UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

Sistema de recomendación de recursos basado en filtrado colaborativo para la plataforma edX

PROYECTO FIN DE CARRERA:

INGENIERÍA TÉCNICA DE TELECOMUNICACIONES,
ESPECIALIDAD EN TELEMÁTICA

- Autora: **Alma Collado Sánchez**
- Tutor: **Pedro José Muñoz Merino**

Leganés, Octubre de 2014

“Para entender el corazón y la mente de una persona no te fijes en lo que ha hecho, no te fijes en lo que ha logrado, sino en lo que aspira a hacer.”

Jalil Gibran

Agradecimientos

A mi abuelo, que aunque nunca se lo dije siempre ha sido mi principal fuerza y motivación. Al final lo he conseguido. Gracias por sentirte orgulloso de mí.

A mis padres, que me han dado la oportunidad de estudiar y me han apoyado durante todos estos años. Gracias por haberme educado y haber sabido transmitirme tan buenos valores y vuestro conocimiento sobre las cosas importantes de la vida.

A mi hermano, que ha decidido seguir mis pasos, gracias por ver en mi un ejemplo a seguir, por tu ayuda y comprensión todos estos meses, por escucharme y estar siempre aquí. Espero darte fuerza para que continúes y alcances esta meta, y que sea la primera de muchas.

A toda mi familia y amigos, por haberse alegrado con mis alegrías y hacer que las penas no lo fueran tanto. Gracias por vuestro apoyo.

A todos mis profesores y compañeros, y en general, a todos aquellos que en algún momento han formado parte de mi vida. Gracias por dejarme aprender de vosotros, tanto de lo bueno como de lo malo.

A Pedro J. Muñoz Merino, por su ayuda y dedicación durante estos meses. Gracias por tus ideas y por haberme dado la oportunidad de trabajar contigo.

Gracias a todos.

Resumen

En los últimos años ha surgido el concepto de MOOC (Massive Open Online Course) así como plataformas asociadas para dar soporte a estos cursos. Las plataformas diseñadas para este fin ofertan cursos en línea accesibles a nivel mundial de manera abierta, por lo que tienen que afrontar el reto de dar servicio a un gran número de usuarios, del orden de miles o decenas de miles, de forma que su experiencia de aprendizaje sea óptima y lo más productiva posible. Una de las plataformas de MOOCs más populares es edX.

En este proyecto se ha desarrollado un recomendador de recursos educativos para la plataforma edX, incluyendo el diseño de un algoritmo de recomendación de los problemas que el usuario debería realizar a continuación. El algoritmo se basa en recomendar problemas con los que han interaccionado, realizado y aprobado usuarios similares al usuario al que se va a realizar la recomendación. La similitud de los usuarios se calcula en función de las notas obtenidas en los distintos problemas del curso. En cualquier momento el usuario puede seleccionar la pestaña de recomendación (*Recommend Me!*), situada en la parte superior del sistema de gestión del aprendizaje (LMS) junto al resto de las pestañas del curso, y obtener un número determinado de problemas propuestos.

Para el desarrollo de esta aplicación se ha trabajado con una instancia de la plataforma edX y se ha utilizado el *framework Django*. Posteriormente, para la evaluación del algoritmo se han creado dos cursos ficticios en el sistema de gestión de contenidos (CMS) de la plataforma e, igualmente, registrar de forma ficticia varios alumnos en estos cursos para hacer una serie de pruebas y poder garantizar la precisión de las recomendaciones generadas por el algoritmo.

Palabras clave: e-learning, MOOC, edX, recomendador, algoritmo de recomendación, *Django*, CMS y LMS.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación

La aparición de los MOOCs (*Massive Open Online Courses*) ha supuesto una gran revolución en la educación, un gran número de universidades e instituciones quieren ofrecer sus cursos en abierto de manera masiva. Sin embargo, en los MOOCs, no es factible que un profesor pueda dar una ayuda y guía personalizada debido al elevado número de alumnos. Es por ello que se hace evidente la necesidad de crear mecanismos automáticos como los recomendadores para dar esa ayuda y guía personalizada a los alumnos.

Algunas plataformas importantes de MOOCs actuales ya incluyen recomendadores, por ejemplo Coursera, sin embargo, no podemos conocer su funcionamiento debido a que no es una plataforma de código abierto. La plataforma edX no tiene ningún sistema de recomendación actualmente.

Por otro lado, los sistemas de recomendación están cada vez más presentes en nuestra vida virtual diaria, y, en concreto, los sistemas de recomendación aplicados a la educación (TEL) están siendo objeto de numerosos estudios [55], intentando que su inclusión pueda mejorar el aprendizaje.

La plataforma edX es una plataforma en continua evolución gracias a su proyecto de código abierto Open edX. En este proyecto colaboran desarrolladores de todo el mundo introduciendo nuevas funcionalidades con el fin de convertir edX en una plataforma potente y accesible. El poder mejorar esta plataforma mediante un recomendador que facilite el aprendizaje es la motivación fundamental de este proyecto.

Estos factores son los que han impulsado y permitido la creación de una herramienta para dicha plataforma que se encargue de proponer los problemas adecuados al nivel de cada alumno en función de su evolución a lo largo del curso, al igual que haría un profesor. Se ofrece así una enseñanza más personalizada que se adapta a las distintas necesidades y proporciona al alumno una experiencia educativa de calidad.

También, al ser edX una plataforma relativamente nueva, la creación de una aplicación desde cero supone un gran reto ya que carece de la documentación necesaria y hay que recurrir a la ayuda de otros colaboradores o de los propios desarrolladores de la plataforma, como he tenido que hacer en varias ocasiones a lo largo del proyecto. Esta relación de colaboración supone una motivación adicional, la de pertenecer a la comunidad edX.

1.2. Objetivos

El objetivo final de este proyecto es la creación de una aplicación recomendador de recursos en Python integrada en el propio código de la plataforma virtual edX que aparezca como una pestaña adicional dentro de cada curso. Seleccionando esta pestaña, los alumnos tendrán acceso a sus problemas recomendados para dicho curso en cualquier momento. Un algoritmo de recomendación será el encargado de proponer estos problemas en función de las notas obtenidas en los problemas realizados por el alumno hasta ese momento. Mediante el cálculo de la similitud con el resto de los compañeros se hará una estimación de los problemas más convenientes para el alumno. La idea general es recomendar problemas que han hecho bien otros alumnos que tienen una similitud en sus interacciones con dicho alumno, en el sentido de que han realizado correctamente los mismos problemas. Se intentará que las propuestas sean lo más precisas posibles para ofrecer una recomendación fiable y de calidad.

Para lograr ese objetivo general, se han tenido que realizar un conjunto de sub-objetivos. El primer sub-objetivo fue entender el funcionamiento y la estructura de la plataforma edX, haciendo un estudio del código con la ayuda de la documentación así como aprender a utilizar el *framework Django* para poder realizar un esquema inicial de los pasos a seguir. También se hizo un análisis exhaustivo de la base de datos *MongoDB* donde se almacena toda la información relacionada con el curso. Posteriormente, se creó un modelo de *Django* que se corresponde con una tabla en la base de datos *MySQL* donde se almacenarán los problemas que se van a recomendar al alumno.

Como segundo sub-objetivo se encuentra la recuperación de la información de las tablas de la base de datos *MySQL*, de donde se obtienen las notas de los problemas del curso realizados por el alumno hasta ese momento así como los identificadores de sus

compañeros de curso, que serán utilizados como parámetro de entrada en el algoritmo de recomendación.

El tercer sub-objetivo fue la creación del algoritmo de recomendación. Una vez desarrolladas las funciones para extraer la información necesaria de las bases de datos se diseñó el algoritmo encargado de calcular las similitudes con los compañeros del curso y obtener los problemas a recomendar. Una idea general del funcionamiento del algoritmo es la siguiente:

1. Se seleccionan los compañeros que tienen el mayor número de problemas realizados y aprobados similares al alumno al que se va a hacer la recomendación.
2. De entre esos compañeros más coincidentes se seleccionan aquellos que difieren en menor número de problemas realizados y aprobados con el alumno al que se va a hacer la recomendación.
3. Se obtienen los problemas aprobados por los compañeros más coincidentes y menos diferentes que no haya realizado el alumno actual.
4. De entre esos problemas se buscan los más populares, es decir los aprobados por el mayor número de compañeros. Serán los problemas propuestos para el alumno actual en ese momento.

El cuarto sub-objetivo fue la realización de pruebas y evaluación del algoritmo para comprobar su correcto funcionamiento.

1.3. Planificación

Los pasos llevados a cabo para la consecución de los objetivos indicados en el apartado anterior son los siguientes:

1. Instalación de una instancia de la plataforma edX y generar el entorno de trabajo instalando las herramientas necesarias.
2. Aprendizaje de programación en Python.
3. Estudio del código de la plataforma y lectura de la documentación disponible.

4. Documentación sobre *Django* y realización del Tutorial de creación de una aplicación en *Django*.

5. Creación de un usuario y contraseña de administración para el CMS (Studio) (sistema de gestión de contenido, del inglés *Content Management System*) de la plataforma edX. Familiarización con el CMS y creación de un curso de ejemplo con esta herramienta de autoría con varias secciones, subsecciones y problemas de cada tipo.

6. Acceso al LMS (sistema de gestión de aprendizaje, del inglés *Learning Management System*) con el usuario creado anteriormente. Familiarización con el LMS.

7. Acceso a las bases de datos *MongoDB* y *MySQL,* para estudiar la forma de almacenamiento de la información introducida en los dos puntos anteriores: al crear cursos y al interaccionar con los cursos creados

8. Registro en la plataforma de varios usuarios ficticios. Darlos de alta en el curso creado y acceder al CMS como administrador para cambiar su estado a *Activo* (al ser usuarios ficticios la dirección de email introducida es también ficticia por lo que no reciben el correo de activación y es el administrador el que debe cambiarlo de forma manual en el CMS).

9. Creación de un modelo de *Django* que se corresponde con una tabla en la base de datos *MySQL* para almacenar cada alumno y sus problemas propuestos.

10. Aprender a obtener la información almacenada en la base de datos *MongoDB.*

11. Entrar en el LMS con los distintos usuarios registrados en el curso y realización de varios problemas con cada uno de ellos a distintos niveles, es decir, que unos vayan más avanzados que otros y aprueben o suspendan distintos problemas para probar el algoritmo de recomendación de forma exhaustiva.

12. Documentación y estudio de los distintos tipos de algoritmos de recomendación y elección del más apropiado.

13. Diseño e implementación del algoritmo de recomendación.

14. Integración del código en la plataforma, inclusión de la pestaña de recomendación (denominada *Recommend me!*) y creación del fichero *HTML* para que la apariencia sea los más similar posible a la del resto de la plataforma.
15. Diseño de los casos de prueba necesarios para cubrir todas las opciones y comprobación del funcionamiento del algoritmo para cada uno de los casos.
16. Documentación del proyecto.
17. Redacción de la memoria.

Para algunos de los pasos anteriores ha sido necesaria la ayuda o guía de los propios desarrolladores de la plataforma mediante el grupo de Google, intercambio de emails o videoconferencia.

2. Estado del arte

En este capítulo se exponen los trabajos relacionados con el presente proyecto, se explican los sistemas que han impulsado su desarrollo y las teorías subyacentes.

2.1. e-Learning

2.1.1. Definición

Aunque existen diferentes concepciones del término e-learning, en este trabajo consideraremos e-learning (también llamado aprendizaje electrónico, aprendizaje en red, teleformación o aprendizaje virtual) como la educación a distancia que utiliza la red como tecnología de distribución de los contenidos. Como soporte de este proceso de enseñanza-aprendizaje se emplean herramientas de hipertexto (correo electrónico, páginas web, foros de discusión, mensajería instantánea y plataformas de formación).

Podemos decir que el e-learning es un sistema basado en una comunicación masiva y bidireccional con el objetivo final de mejorar el aprendizaje.

Existen muchas definiciones formales del concepto e-learning, por ejemplo, en un informe de la Dirección General de las Telecomunicaciones de Teleeducación [1] se dice que es "*una enseñanza a distancia, abierta, flexible e interactiva basada en el uso de las nuevas tecnologías de la información y de la comunicación (TIC), y de las comunicaciones, y sobretodo aprovechando los medios que ofrece la red Internet*"[1]. O más en detalle, como se expone en Jolliffe et al. (2001) [2], "*puede ser descrito como la distribución y el acceso a colecciones coordinadas de materiales de aprendizaje sobre un medio electrónico usando un servidor web para distribuir los materiales, un navegador web para acceder a ellos y los protocolos TCP/IP y HTTPP para mediar el intercambio.*

Este proyecto ha sido desarrollado para ser integrado en una de estas plataformas de formación, concretamente edX, con el objetivo de personalizar y facilitar el aprendizaje

[1] http://www.edudistan.com/ponencias/Arturo%20Azcorra%20Salona.htm

de los alumnos mediante la recomendación de contenido en base a sus calificaciones a lo largo del curso.

2.1.2. e-Learning vs formación presencial

TABLA 2-1. CARACTERÍSTICAS DEL E-LEARNING Y DE LA FORMACIÓN PRESENCIAL (SACADA DE [3])

e-Learning	Formación presencial
Permite que los estudiantes vayan a su propio ritmo	Parte de una base de conocimiento y el estudiante debe adaptarse a ella
Es una formación basada en el concepto de *formación en el momento en que se necesita (just in time)*	Los profesores determinan cuándo y cómo los estudiantes recibirán los materiales formativos
Permite la combinación de diferentes materiales (auditivos, visuales y audiovisuales)	Parte de la base de que el sujeto recibe pasivamente el conocimiento para generar actitudes innovadoras, críticas e investigadoras
Con una sola aplicación puede atenderse a un mayor número	Tiende a apoyarse en materiales impresos y en el profesor como fuente de estudiantes de presentación y estructuración de la información
El conocimiento es un proceso activo de construcción	Tiende a un modelo lineal de comunicación
Tiende a reducir el tiempo de formación de las personas	La comunicación se desarrolla básicamente entre el profesor y el estudiante
Tiende a ser interactiva, tanto entre los participantes en el proceso (profesor y estudiantes) como con los contenidos	La enseñanza se desarrolla de forma preferentemente grupal
Tiende a realizarse de forma individual, sin que ello signifique la renuncia a la realización de propuestas colaborativas	Puede prepararse para desarrollarse en un tiempo y en un lugar
Puede utilizarse en el lugar de trabajo y en el tiempo disponible por parte del estudiante	Se desarrolla en un tiempo fijo y en aulas específicas
Es flexible	Tiende a la rigidez temporal
Tenemos poca experiencia en su uso	Tenemos mucha experiencia en su utilización

No siempre disponemos de los recursos estructurales y organizativos para su puesta en funcionamiento	Disponemos de muchos recursos estructurales y organizativos para su puesta en funcionamiento

Como podemos ver en la tabla 2-1 tomada de [3], el e-learning parece tener muchas ventajas con respecto a la formación presencial, flexibiliza y amplía la información, permite la deslocalización del conocimiento, facilita una formación colaborativa, ahorra costes y desplazamiento, etc., pero, también posee inconvenientes, por ejemplo, requiere más inversión de tiempo por parte del profesor, precisa unos conocimientos tecnológicos tanto por parte del profesor como de los alumnos, todavía no se tiene mucha experiencia en su utilización, depende de una conexión a Internet, etc. Sin embargo, algunos de estos problemas dejarán de serlo con el paso del tiempo, el e-learning cada vez más se está convirtiendo en algo ordinario y, conforme se vaya extendiendo el uso de la enseñanza en red en nuestro sistema educativo aumentará el nivel de experiencia.

2.1.3. Criterios para una enseñanza de calidad

Según Pallof et al. (2003) [4] existen una serie de principios y lecciones a considerar antes de poner en funcionamiento cualquier tipo de acción formativa en la red para garantizar el éxito de buenas prácticas educativas. Esto se resume en la tabla 2-2:

TABLA 2-2. PRINCIPIOS QUE DEBEN CONSIDERARSE PARA LA FORMACIÓN EN LÍNEA (SACADA DE [4])

Principio	Lección
Principio 1. La buena práctica anima al estudiante a tomar contacto	El instructor debe ofrecer guías claras con la facultad para la interacción con los estudiantes
Principio 2. La buena práctica anima la cooperación entre los estudiantes	Una discusión bien diseñada facilita significativamente la cooperación entre los estudiantes
Principio 3. La buena práctica facilita un aprendizaje activo	El estudiante debe presentar proyectos durante el curso
Principio 4. La buena práctica implica un feedback rápido	El instructor necesita ofrecer dos tipos de feedback: de información y de acuse (de haber recibido la información)

Principio 5. La buena práctica pone énfasis en el tiempo en la tarea	Los cursos en línea necesitan una fecha tope
Principio 6. La buena práctica comunica elevadas expectativas	Se provocan tareas, ejemplos de caso y alabanzas comunicando la calidad de los trabajos
Principio 7. Las buenas prácticas respetan los diversos talentos	Se permite a los estudiantes que elijan y caminos de aprendizaje los temas de los proyectos y se deja que emerjan diferentes puntos de vista

Para la búsqueda de criterios de calidad del e-learning se creó el Teleobservatorio Universitario de Docencia Virtual (NetLab) [5], un proyecto para la transmisión de experiencias y contenidos sobre docencia virtual cuyo objetivo es favorecer el intercambio de buenas prácticas entre grupos de investigación y profesionales interesados en la incorporación de las TIC al ámbito universitario.

Según Cabero en su artículo "*Bases pedagógicas del e-learning*" [6], dejando al margen la tecnología y la problemática de las plataformas, existen nueve variables que hay que tener en cuenta a la hora de buscar el éxito de las acciones formativas en la red. Estas variables están representadas en la figura 2-1.

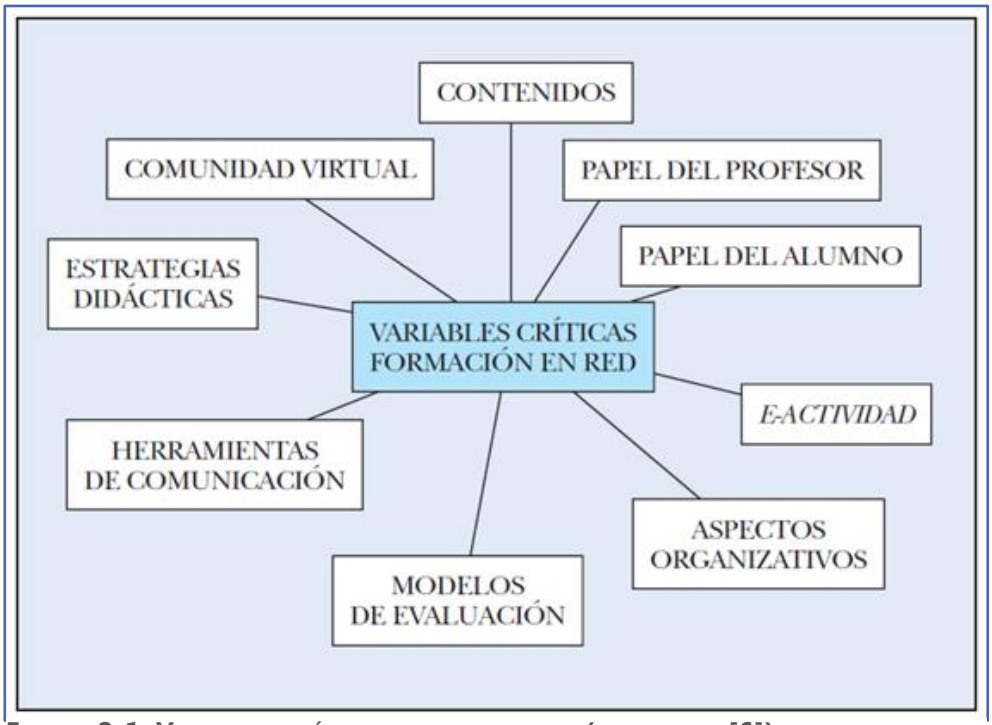


FIGURA 2-1. VARIABLES CRÍTICAS DEL E-LEARNING (SACADA DE [6])

Estas variables están estrechamente relacionadas entre sí y es indispensable atender cada una de ellas para garantizar un proceso de enseñanza-aprendizaje de calidad. Estructurar los contenidos, diseñar actividades, orientar y guiar el aprendizaje, fomentar la comunicación entre los participantes de la comunidad virtual o crear estrategias didácticas para la consecución de los objetivos propuestos, son acciones fundamentales de las que dependerá el éxito o fracaso de la experiencia formativa.

2.1.4. Plataformas de e-learning

Una plataforma de e-learning o sistema de gestión de aprendizaje es un entorno virtual de aprendizaje orientado a facilitar la experiencia de la formación a distancia, tanto para empresas como para instituciones educativas.

Más en detalle, una plataforma de e-learning es según [7] *"un software instalado en un servidor web que se emplea para administrar, distribuir y controlar las actividades de formación no presencial (o aprendizaje electrónico) de una institución u organización"*. Tiene varias funcionalidades, según [8], tales como: *"registra a todos los actores que intervienen en el acto de aprendizaje (alumnos, profesores, administradores, etc.),*

organiza los diferentes cursos en un catálogo, almacena datos sobre los usuarios, realiza un seguimiento del aprendizaje y genera informes automáticamente para tareas de gestión".

Clarenc [9], señala que los LMSs pueden ser de tres tipos:

- **Comercial**: implica el pago de algún tipo de licencia, ya sea a la empresa que desarrolló el sistema o a la que lo distribuye. Son sistemas generalmente robustos y bastante documentados. Suelen presentarse en paquetes de funcionalidades para cubrir las distintas necesidades y presupuestos, es decir, cuanto más completo sea el paquete que se pague más servicios recibirá a cambio. Entre los más conocidos se encuentran Blackboard, WebCT, OSMedia, Saba, eCollege, Fronter, SidWeb, e-ducativa y Catedr@.

- **De Software Libre**: generalmente están desarrollados por instituciones educativas o por personas vinculadas al sector educativo. Algunas de estas plataformas son de tipo '*Open Source*' (de código abierto), lo que establece que son de libre acceso, permitiendo que el usuario pueda manipular el software, es decir, que una vez obtenido se pueda usar, estudiar, cambiar y redistribuir libremente. Algunas de estas plataformas superan en funcionalidades a las comerciales, mientras que otras sólo cuentan con funcionalidades básicas. Entre las más usadas están ATutor, Dokeos, Claroline, dotLRN, Moodle, Ganesha, ILIAS y Sakai.

- **En la nube**: no son consideradas plataformas de e-learning propiamente dichas. Su mayor utilidad es la de apoyar a la clase presencial, así como el desarrollo de MOOCs. Las más populares son Udacity, Coursera, Udemy, edX, Ecaths, Wiziq y Edmodo, entre otros.

De los MOOCs hablaremos más en detalle en el apartado 2.2, donde se hará un repaso a su historia, analizando su presente, pasado y futuro, se introducirán los distintos tipos de MOOCs y se tratarán los temas tanto de su financiación como de su impacto en el ámbito de la enseñanza.

Más adelante, en el apartado 2.3, se analizará a fondo el proyecto edX, así como su plataforma, para la que se ha realizado el recomendador de recursos descrito en este documento.

2.2. MOOCs

FIGURA 2-2. LOGO DE MOOC

2.2.1. ¿Qué es un MOOC?

Hay muchas definiciones de lo que es un MOOC pero, a grandes rasgos, podemos considerar MOOC a todos aquellos cursos que cumplen los siguientes criterios básicos:

- ***Massive***: Pueden acceder a ellos un gran número de estudiantes. Mucho mayor que si se impartieran en un aula y usualmente mayor a los que acceden en cursos privados tradicionalmente soportados por los LMSs.

- ***Open***: No requieren conocimientos previos ni registro o suscripción en una institución. El acceso es inmediato, sin restricciones a material digital educativo, académico, científico o de cualquier otro tipo, son de acceso abierto. Sin embargo, las plataformas pueden ofrecer servicios complementarios de carácter no gratuito.

- ***Online***: El contenido se distribuye de forma online aprovechando todos los recursos que ofrece la web (videos, audios, textos, animaciones). Y facilita la interacción asíncrona entre los participantes.

- ***Course***: Sigue un plan de estudios orientado a la adquisición de unos conocimientos específicos o a la consecución de unos objetivos.

Algunas de las propiedades anteriores son compartidas con otro tipo de recursos y puede llevar a confusión a la hora de caracterizarlos o no como MOOCs, a continuación se citan algunos ejemplos tomados de [10]:

- Las recopilaciones de material didáctico, como Khan Academy, Open Courseware o iTunes U, son masivas y online pero no conforman un curso como tal.
- Series de conferencias sin un resultado (LSE Public Lectures, New Books Network) son online y de acceso abierto pero tampoco son un curso y puede que no sean masivas.
- Una red de aprendizaje o *Personal Learning Network* es online y abierta pero no es masiva ni es un curso.
- Una comunidad en línea de apoyo al estudio es online pero no es un curso y puede que no sea ni masiva ni abierta.
- Las conferencias y seminarios ofrecidos en vivo a gran escala (*The Reith Lectures*) son masivas y pueden parecer un curso pero la interacción entre los participantes es limitada.

2.2.2. Orígenes

La aparición de los MOOCs está ligada a otros dos fenómenos que han tenido lugar principalmente durante los últimos diez años:

- **Los Recursos Educativos Abiertos (*Open Educational Resources*):** desde que en 1999 el MIT lanzase su proyecto OpenCourseWare (OCW) muchas instituciones han creado sus propias versiones del mismo y puesto a disposición del público los contenidos de muchas asignaturas de sus programas de estudio. Esto ha permitido el acceso masivo y abierto a los contenidos y recursos ofrecidos en el curso.
- **El aprendizaje social abierto (*Open Social Learning*)**: la aparición de la Web 2.0 ha propiciado que los usuarios se conviertan en protagonistas del proceso. Aportan, colaboran e interactúan en la red dando lugar a un conocimiento compartido.

Es en ese momento, una vez que las universidades ya disponen de los contenidos abiertos y de la tecnología apropiada sin tener que realizar grandes inversiones sin un claro retorno económico directo, cuando empiezan a aparecer los primeros MOOCs.

2.2.3. ¿Cuál fue el primer MOOC?

Los primeros MOOCs aparecieron alrededor del año 2001, cuando las Universidades de Oxford, Yale y Stanford se unieron en un proyecto sin ánimo de lucro, **AllLearn** (Alliance for Lifelong Learning), y comenzaron a ofrecer cursos de interés general, inicialmente para los alumnos de dichas universidades y posteriormente, ya en el año 2002, abiertos a todo el público [11].

Su modelo de negocio se basaba en un enfoque flexible y tecnológico y contaba con la participación de los mejores profesores y expertos de las tres universidades. Sin embargo, desde el principio, no se cumplieron los objetivos de matriculación.

A diferencia de otros proveedores de aprendizaje online, AllLearn era una empresa sin fines de lucro, sin socios comerciales importantes o inversores privados. Las tasas de matriculación se establecieron inicialmente en $195 para los alumnos y $250 para el resto del público, además de otros gastos variables entre $11.95 y $49.95. Producir un curso costaba entre $10.000 y $150.000, por lo que se requería una matriculación de entre 40 y 600 estudiantes por curso para alcanzar el equilibrio. Los fundadores del Proyecto subestimaron los costes del diseño de los cursos y sobreestimaron el número de estudiantes dispuestos a pagar los costes de la matrícula.

Cinco años después, la empresa anunció que el proyecto no era sostenible. Cesó sus operaciones a finales de 2005 para disolverse, finalmente, en 2006 [12]. AllLearn llegó a ofrecer 110 cursos a más de 10.000 estudiantes de 70 países.

En la misma línea de AllLearn, a finales de 2000 aparece **Fathom.com** [13], un portal de aprendizaje online encabezado por la Universidad de Columbia con socios como The London School of Economics and Political Science, the British Library, the New York Public Library, Cambridge University Press, and the Smithsonian Institution's National Museum of Natural History.El proyecto, más allá de limitarse a los materiales del curso, incluía una amplia gama de contenidos educativos multimedia (de ahí su asociación con instituciones de archivos), herramientas de aprendizaje colaborativo y grupos de debate dirigidos por expertos.

Aunque Fathom ofrecía muchos seminarios gratuitos, algunos cursos costaban más de $500. En 2002, ajustaba, sin éxito, su modelo de negocio para generar ingresos adicionales pero la crisis económica impidió que la empresa con fines de lucro alcanzara sus objetivos y cerró a principios de 2003 con una participación de alrededor de 65.000 estudiantes de 52 países.

Tras estos dos proyectos fallidos, varias fuentes coinciden en que el primer MOOC fue '*Conectivism and Connective Knowledge'*, organizado por George Siemens y Stephen Downes (University of Manitoba, Canadá) en agosto de 2008. El curso duró 12 semanas y alcanzó las 2.300 inscripciones. Según el artículo de George Siemens '*What is the theory that underpins our moocs?'* [14] fue en ese momento cuando Dave Cormier y Bryan Alexander inventaron el término MOOC.

Sin embargo, existen otros cursos anteriores que también podrían ser considerados MOOCs:

- ***'From NAND to Tetris: Building a Modern Computer from First Principles'***, organizado por Noam Nisan (Hebrew University de Jerusalem, Israel) y Shimon Schocken (IDC Herzliya, Israel) en 2005[2].
- ***'Introduction to Open Education'***, organizado por David Wiley (Utah State University) en 2007[3].
- ***'EC&I 831: Social Media & Open Education'***, organizado por Alec Couros (University of Regina, Canadá) por primera vez en enero de 2008[4].

Según los estándares actuales, el primer curso en alcanzar el carácter masivo de los MOOCs fue *'Introduction to Artificial Intelligence'* [5], organizado por Sebastian Thrun y Peter Norvig en octubre de 2011, que contó con la participación de 160.000 alumnos de todo el mundo. Posteriormente, tras el éxito de este primer curso, Sebastian Thrun se dio cuenta del potencial de esta nueva forma de enseñanza y dejó su puesto como profesor en la Universidad de Stanford para fundar la plataforma Udacity [15].

[2] http://www.ted.com/talks/shimon_schocken_the_self_organizing_computer_course
[3] ocw.usu.edu/instructional-technology-learning-sciences/introduction-to-open-education/index.html
[4] http://couros.ca/cv/teaching/innovation-in-teaching-learning/
[5] https://www.udacity.com/course/cs271

El siguiente curso de gran éxito fue '*Circuits & Electronics*'[6] en la primavera de 2012, organizado por el profesor Anant Agarwal del Massachusetts Institute of Technology (MIT) en su plataforma MITx, alcanzó las 120.000 inscripciones.

Estos dos éxitos inesperados fueron el detonante de la actual atención que reciben estos cursos. Poco después, los profesores Andrew Ng y Daphne Koller, también de la Universidad de Stanford, fundaron la plataforma Coursera y comenzaron a ofrecer cursos a partir de abril de 2012.

Posteriormente, el 2 de mayo de 2012, el MIT y la Universidad de Harvard, anunciaron su proyecto conjunto edX, cuyo objetivo era desarrollar una plataforma común sin ánimo de lucro. Desde entonces, ambas instituciones junto con otras que se han ido uniendo posteriormente al proyecto, ofrecen cursos gratuitos a través de Internet en un proyecto colaborativo que busca romper los moldes de la educación universitaria tradicional.

2.2.4. Tipos de MOOCs

Hay varias formas de clasificar los MOOCs, la más conocida distingue dos tipos principales [16] y un tercer tipo híbrido no reconocido por muchos (Ver tabla 2-3 extraída de [19]):

- **cMOOCs o MOOCs conectivistas**: fueron los primeros en aparecer ('*Introduction to Open Education*', '*Connectivism and Connective Knowledge*') y se caracterizan por dar prioridad a la creación de conocimiento por parte de los estudiantes, la creatividad, la autonomía, y el aprendizaje social y colaborativo.
- **xMOOCs o MOOCs comerciales**: son los que se han hecho más populares, se ofrecen a través de plataformas comerciales o semicomerciales como Coursera, edX y Udacity. Se caracterizan por ofrecer un aprendizaje tradicional, basado en la visualización de videos y la realización de ejercicios.

[6] cw.mit.edu/courses/electrical-engineering-and-computer-science/6-002-circuits-and-electronics-spring-2007/

- **tMOOCs o MOOCs híbridos**: son una mezcla de los dos tipos anteriores. Se centra en la realización de tareas y actividades por parte del estudiante a través de las cuales va avanzando en el curso [17, 18]. Estas tareas incluyen resolución de casos, lectura y análisis de documentos, resolución de problemas de forma individual o grupal, etc.

TABLA 2-3. COMPARACIÓN DE LOS TIPOS DE MOOCS (SACADA DE [19])

	cMOOC	xMOOC	tMOOC
Teoría de aprendizaje	Conectivismo	Conductismo	Híbrida
Modelo de aprendizaje	Co-generación de conocimiento	Adquisición (clásica) de conocimiento	Adquisición y generación de conocimiento
Orientado a ...	Redes	Contenidos	Tareas
Conocimiento	Distribuido y co-generado	Declarado por la organización del curso	Declarado por la organización del curso
Contenido	Co-creación o socio-creación a partir de un tema central	Tema central; desarrollo enfocado a conceptos	Tema central; desarrollo enfocado a actividades
Objetivos del aprendizaje	Autodefinido por cada participante	Definido por el profesor	Definido por el profesor
Ámbito del aprendizaje	Tema central abierto; interdisciplinar	Tema central cerrado; disciplinar	Tema central cerrado; disciplinar
Interacción	Fuerte	Débil	Media
Cohesión y control del curso	Participantes	Profesor	Profesor
Evaluación	Muy difícil o inexistente	Sencilla; automátizada en múltiples casos	Difícil a muy difícil
Preponderancia de ...	Exploración y comunidad sobre los contenidos	Adquisición de contenidos sobre la comunidad	Tareas sobre la comunidad

Otra clasificación organiza los MOOCs en tres grupos en función de tres elementos clave para su funcionamiento (ver figura 2-3 extraída de [20]):

- **Network-based:** los primeros MOOCs pertenecen a este grupo. Se centran en las relaciones que se establecen entre los participantes de los cursos. No se puede utilizar la evaluación tradicional.

- **Task-based:** lo más importante es la adquisición de ciertas aptitudes y destrezas mediante la realización de actividades. La creación de una comunidad de alumnos es importante pero no es lo principal.
- **Content-based:** lo más importante es la asimilación del contenido. La creación de una comunidad de alumnos es secundaria y un alumno puede superar el curso sin relacionarse con el resto de los alumnos. Se emplea la evaluación tradicional mediante ejercicios tipo test debido al enorme número de alumnos que pueden inscribirse en un curso de este tipo.

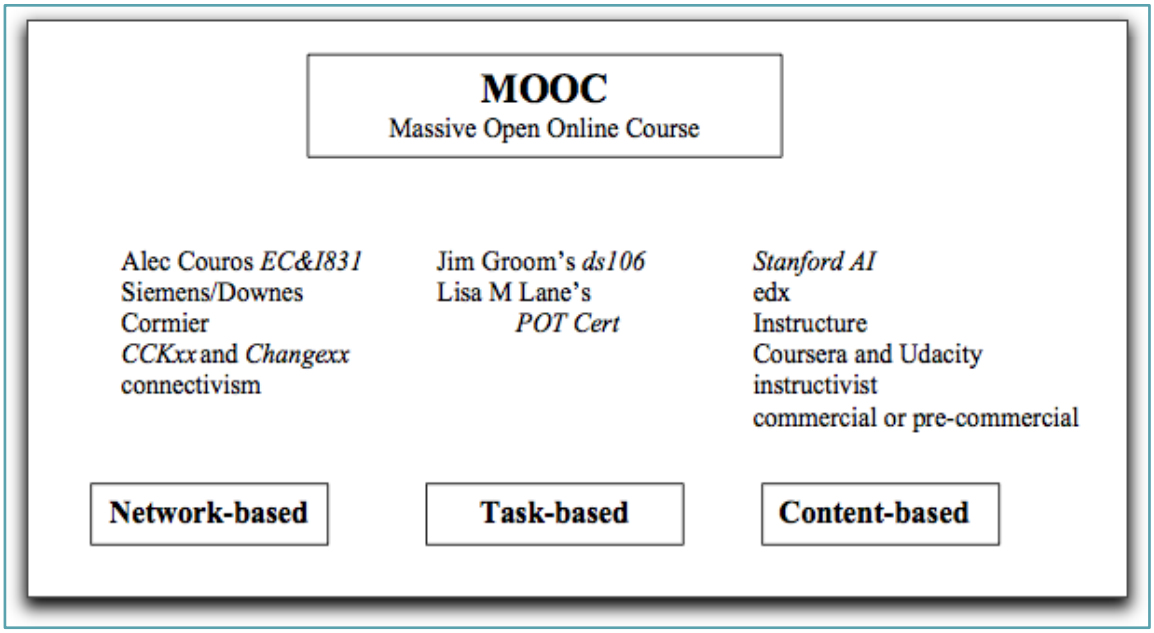


FIGURA 2-3. CLASIFICACIÓN DE LOS MOOCS EN FUNCIÓN DE LOS ELEMENTOS CLAVE PARA SU FUNCIONAMIENTO (SACADA DE [20])

2.2.5. La financiación de los MOOCs

Los proyectos más famosos están financiados por las propias universidades organizadoras, aunque hay otros proyectos en los que también participan empresas con inversiones de capital riesgo.

Muchas empresas están interesadas en financiar este tipo de proyectos con el único fin de buscar talento, tener acceso a los datos de los mejores estudiantes de todo

el mundo es tener información privilegiada. Se amplía el campo de búsqueda, los mejores ya no solo se encuentran entre los alumnos de las universidades más elitistas, un reducido grupo que paga por recibir conocimiento, sino que se abre a todo el público.

Sin embargo, que los cursos actualmente sean gratuitos no significa que en el futuro lo vayan a seguir siendo, se podrán aplicar tasas ajustadas a los ingresos de los estudiantes. Y, como ya sucede en la actualidad en algunos MOOCs, se ofrecerá la certificación como un servicio de pago. Cuando un alumno quiere certificar su aprendizaje, edX, Coursera o Udacity, por un módico precio y, en ocasiones, previo examen, ya facilitan un certificado personalizado con reconocimiento oficial, podemos ver un ejemplo en la figura 2-4.

FIGURA 2-4. EJEMPLO DE CERTIFICADO VERIFICADO DE EDX

2.2.6. El impacto de los MOOCs

El impacto de los MOOCs se mide en términos temporales, es decir, es necesario realizar un análisis y calcular el tiempo que tardaría un profesor en dar clase de forma presencial al mismo número de alumnos registrados en un MOOC. En un estudio realizado por Sebastian Thrun [21], fundador de Udacity, se hicieron los cálculos para algunas de las universidades más prestigiosas, mostradas en la figura 2-5, y los resultados obtenidos muestran una clara ventaja temporal.

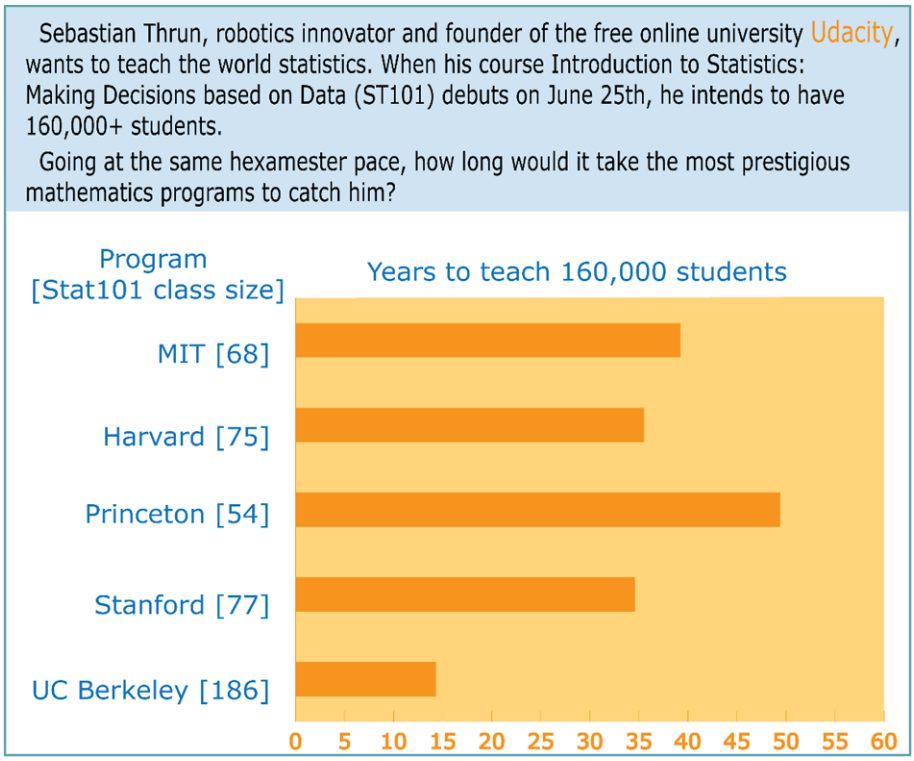


FIGURA 2-5. COMPARACIÓN DEL TIEMPO QUE SE TARDARÍA EN DAR EL CURSO DE FORMA PRESENCIAL EN VARIAS UNIVERSIDADES (SACADA DE [21])

Sin embargo, los resultados que observamos en la figura 2-5 no son exactos ya que no todos los alumnos que se registran en un curso consiguen finalizarlo, esto se debe a que existe una alta tasa de abandono. Según el artículo *MITx - the Fallout Rate* [22], donde se analiza la variación del número de estudiantes a lo largo del primer curso ofrecido por la plataforma edX, 6.002x *Circuits and Electronics* del MITx, de los 154.763 alumnos que se apuntaron, solo 7.157 obtuvieron el certificado, lo cual supone tan solo un 5% de éxito.

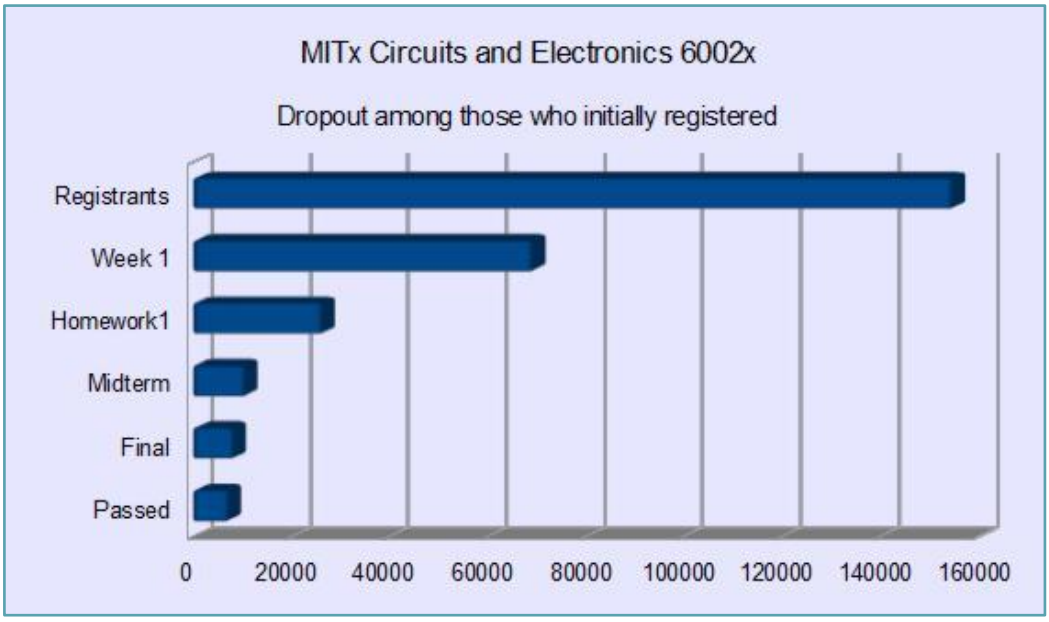


FIGURA 2-6. TASA DE ABANDONO A LO LARGO DEL CURSO (SACADA DE [21])

En el gráfico mostrado en la figura 2-6 observamos que más de la mitad de los estudiantes no llegaron a ver el primer ejercicio, por lo que se puede deducir que la mayor parte se registraron simplemente para ver lo que era el MITx, sin ninguna intención de realizar el curso, o vieron que era muy difícil o que requería mucho tiempo y abandonaron.

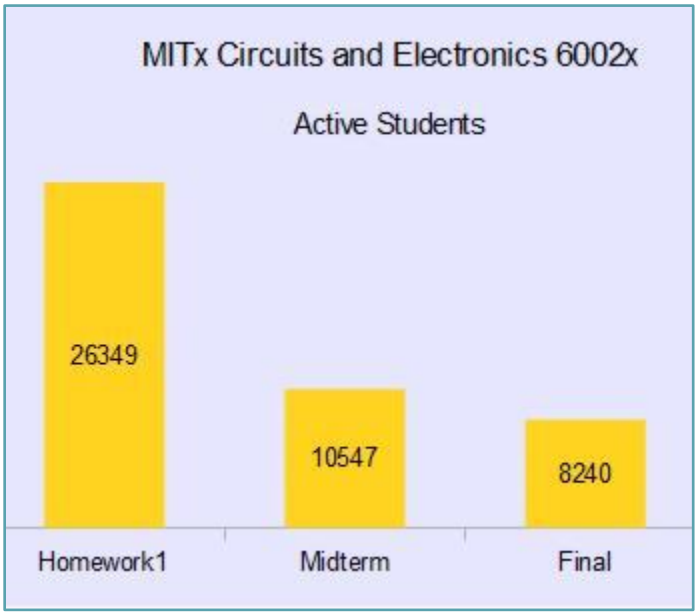


FIGURA 2-7.ESTUDIANTES ACTIVOS A LO LARGO DEL CURSO (SACADA DE [21])

En la figura 2-7 podemos ver que de los 26.349 estudiantes que permanecían activos después de la primera semana, el 60% abandonaron al llegar el examen parcial y otro 22% en el examen final. De estos 8.240 que llegaron al examen final, 7.157 consiguieron el certificado.

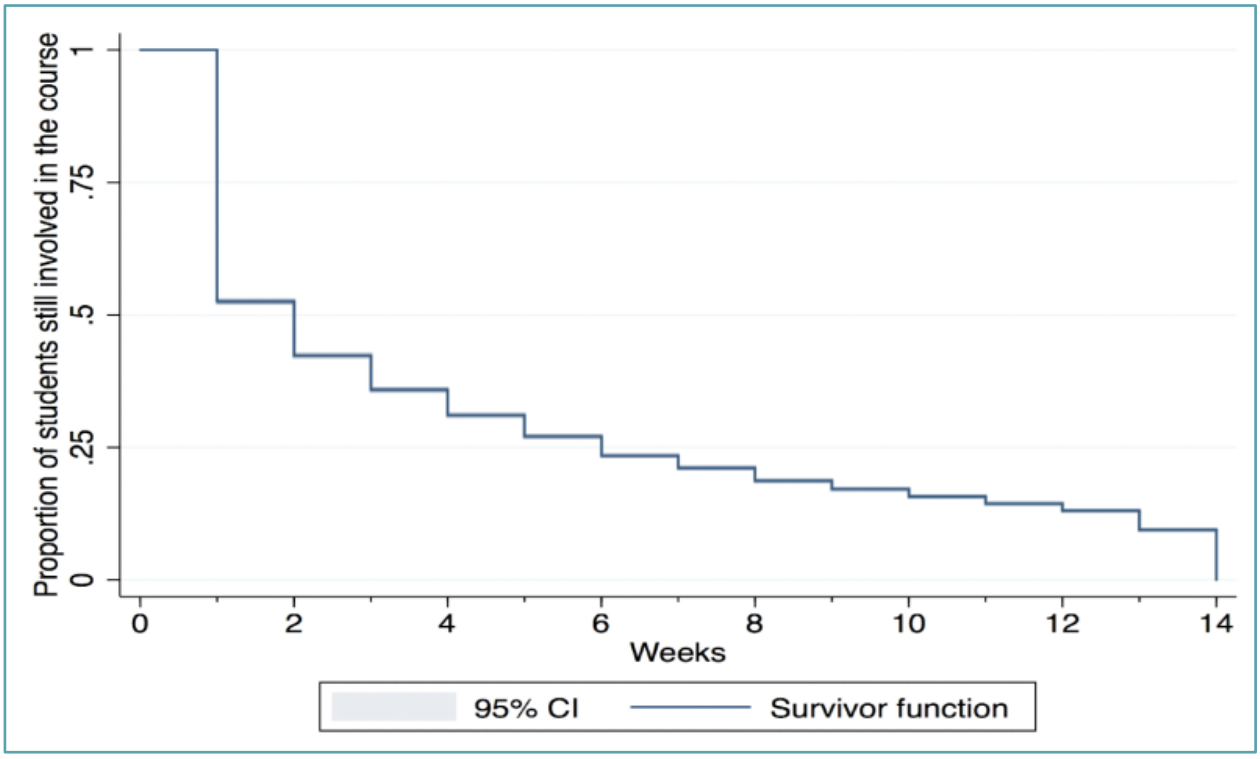


FIGURA 2-8. EVOLUCIÓN DEL PORCENTAJE DE ALUMNOS ACTIVOS A LO LARGO DEL CURSO (SACADA DE [21])

En el gráfico de la figura 2-8 queda patente la disminución del porcentaje de alumnos activos a lo largo del curso. Observamos una disminución progresiva, con una caída muy acusada al final de la primera semana, y que alcanza el 5% en la última semana.

En el artículo *Why Do Students Enroll in (But Don't Complete) MOOC Courses?* [23] se apuntan las posibles causas que llevan a un estudiante a abandonar un MOOC, como simple curiosidad, interés solo en una parte del curso, despreocupación económica de no completar el curso o desinterés por la metodología y/o temática.

En un análisis de 2013 [24] realizado en relación a tres cursos de la plataforma Coursera de distintos niveles de dificultad, '*Computer Science 101*[7]' (*High School level*), '*Algorithms: Design and Analysis*[8]' (*Undergraduate level*) y '*Probabilistic Graphical Models*[9]' (*Graduate level*), aparecen cinco tipos distintos de estudiantes en función de su comportamiento a los largo del curso. Podemos hablar así de un grupo mayoritario de ***No-Shows*** (alumnos que se registraron pero no accedieron al curso en ningún momento), ***Observers*** (alumnos que se limitaron a leer los contenidos o los foros de discusión pero no participaron), ***Drop-Ins*** (alumnos que mostraron actividad en alguna parte del curso pero sin intención de realizarlo entero), ***Passive Participants*** (alumnos que hacen uso del contenido del curso pero no se involucran) y ***Active Participants*** (alumnos cuyo objetivo es completar el curso participando y realizando los ejercicios).

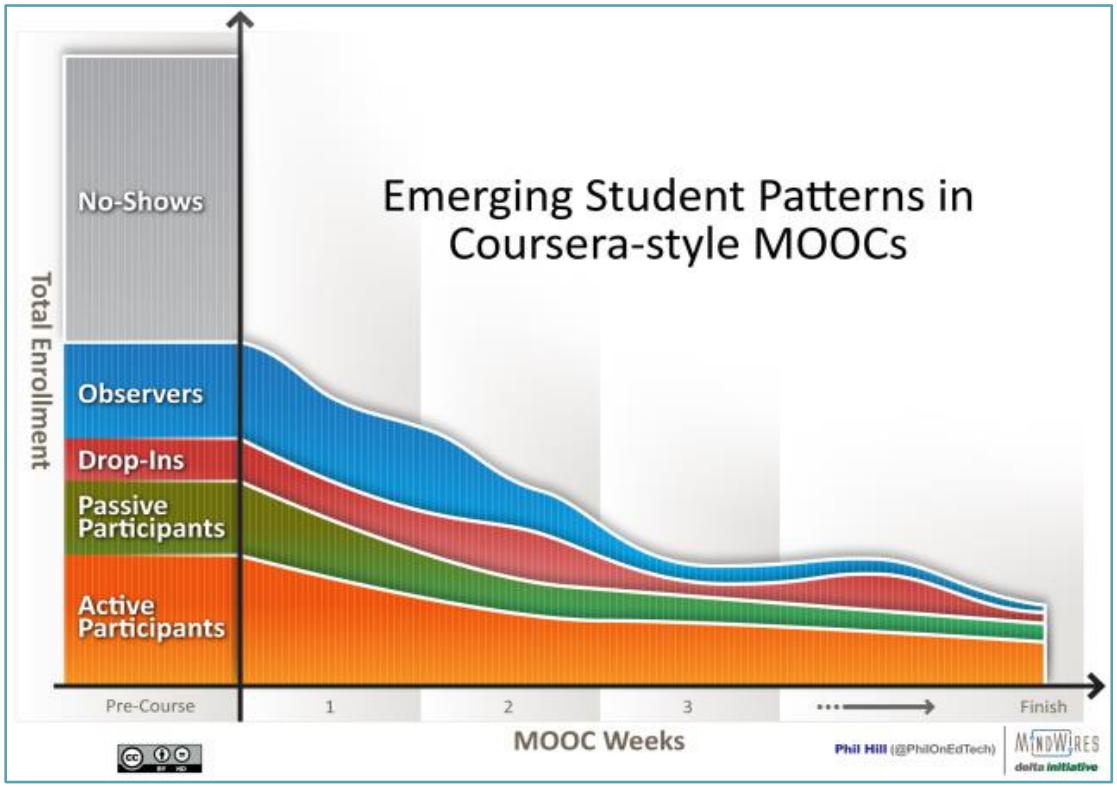


FIGURA 2-9. EVOLUCIÓN DE LOS DISTINTOS TIPOS DE ALUMNOS A LO LARGO DEL CURSO (SACADA DE [24])

[7] https://www.coursera.org/course/cs101
[8] https://www.coursera.org/course/algo
[9] https://www.coursera.org/course/pgm

En el gráfico de la figura 2-9 observamos la disminución del número de cada tipo de estudiantes a lo largo del curso. Vemos cómo la cantidad de alumnos del tipo *Active Participants* es la que menos disminuye, por lo tanto, podríamos decir que los alumnos que se registran con la intención de completar el curso adquieren cierto nivel de compromiso.

Podemos observar que ambos gráficos, el de la figura 2-8 y el de la figura 2-9, siguen el mismo patrón, una disminución progresiva de la participación. Al superponerlos vemos que esa disminución ocurre de forma quasi-equivalente, como se muestra en la figura 2-10.

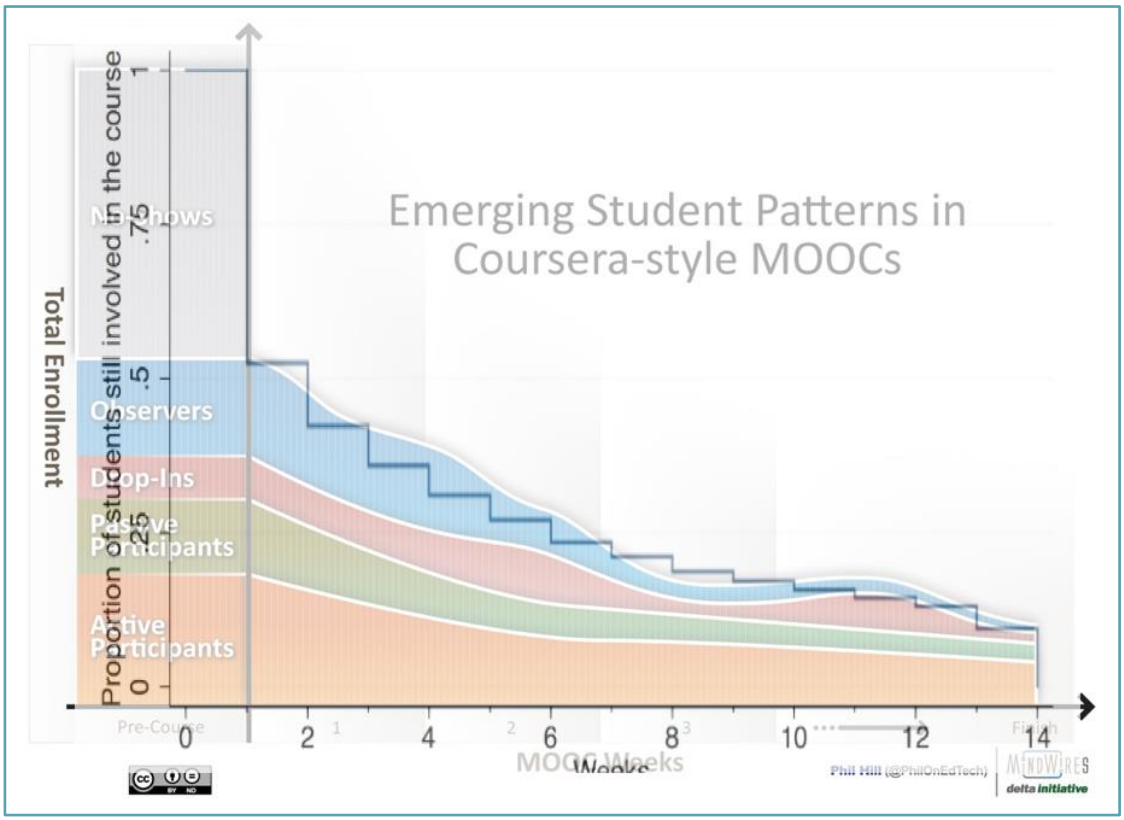


FIGURA 2-10. MISMO PATRÓN DE DISMINUCIÓN (SACADA DE [24])

2.2.7. El futuro de los MOOCs

El futuro de los MOOCs depende, en gran medida, de su valoración, es decir, si las empresas no valoran que sus empleados tengan este tipo de formación se puede convertir en algo temporal.

Los MOOCs no sustituyen ni sustituirán a la Educación Superior, sino que son el complemento perfecto, tienen virtudes que no tienen otro tipo de cursos online pero también carecen de muchas características que poseen los cursos online tradicionales o la formación presencial. Por tanto, podemos decir que son complementarios pero no sustitutivos.

En mayo de 2013, *The Huffington Post* llevó a cabo una encuesta mundial sobre la aceptación de los MOOCs [25]. Estos son algunos de los datos más interesantes:

- El 13% de los centros ya ofrecen MOOCs; en 2016 será el 43%.
- El 72% piensan que los MOOCs son apropiados para continuar la educación superior, el 59% para planificar cursos fuera de los grados y el 53% como fórmulas de entrenamiento.
- Para el 44%, el principal valor de los MOOCs es el desarrollo de nuevas metodologías educativas, para el 35% el incremento de la visibilidad de la institución y para el 16% mejorar la formación de los docentes.
- Para el 41%, el principal problema de los MOOcs es la ausencia de revisiones o de exámene, para el 25% los altos costes de puesta en marcha y para el 15%, la elevada dedicación en tiempo que necesita.
- El 44% de los centros se ha planteado dotar de créditos oficiales sus MOOCs.
- El 83% de los centros se plantean ingresar en plataformas como Coursera o Udacity.
- El 67% de las instituciones creen que los MOOCs nunca podrán sustituir a la educación tradicional y presencial y un 5% estimaron que en 5 años, los cursos abiertos y masivos podrán ser sustitutivos perfectos.

2.3. El proyecto edX

FIGURA 2-11. LOGO DE LA PLATAFORMA EDX

[A partir de este punto me referiré como edX tanto a la organización como a su plataforma de código abierto, OpenEdx, ya que a menudo se tratan como sinónimos]

2.3.1. Descripción

Es una plataforma de código abierto, con licencia Affero GPLv3, y sin ánimo de lucro que ofrece cursos en línea masivos y abiertos (MOOCs) creada en mayo de 2012 por el Instituto Tecnológico de Massachusetts (MIT) y la Universidad de Harvard. EdX cumple un importante papel en la mejora de la investigación en educación y la calidad de la educación en ambos campus. Al mismo tiempo, edX llega a alumnos de todo el mundo a través de sus cursos en línea en una amplia gama de disciplinas de forma gratuita. Comenzó acogiendo contenido de MITx y Harvardx y actualmente 53 escuelas, organizaciones no lucrativas, empresas y organizaciones internacionales ofrecen o tienen previsto ofrecer cursos a través de la plataforma. A 23 de junio de 2014, edX cuenta con más de 2,5 millones de usuarios y más de 200 cursos [26].

Como ya se dijo en el apartado 2.2.2, la creación en 2001 por parte del MIT del OpenCourseWare (OCW), fue uno de los fenómenos que propiciaron la aparición de los MOOCs. Sin embargo, estos cursos no se consideran una enseñanza a distancia ya que al finalizar no se obtiene ninguna certificación. EdX va más allá, permite al estudiante demostrar una cierta maestría en el curso realizado obteniendo un certificado de aprovechamiento (no bajo el nombre del MIT o Harvard) y, además, se da acceso a grupos de discusión, herramientas wiki de aprendizaje colaborativo, laboratorios online e

instrumentos de evaluación para que el alumno pueda conocer su progreso durante el curso e ir marcando su propio ritmo.

Para poder participar en los cursos online de la plataforma edX hay que tener una formación inicial acorde con el nivel del curso al que se quiere acceder (aunque también hay muchos cursos para principiantes), e, inevitablemente, tener conocimientos de inglés, dado que la mayoría se imparten en ese idioma. El proceso para participar es sencillo, registrarse en la plataforma y matricularte en el curso deseado de entre los ofertados.

Centrándonos más en detalle en las funcionalidades que nos ofrece la plataforma podemos destacar las siguientes características [27]:

1. Posibilidad de mostrar lecciones grabadas en vídeo con subtítulos e indexación sobre los propios subtítulos (puedes buscar por palabras que aparezcan en los mismos y al pulsar sobre los resultados, ir directamente a la sección de vídeo que los contiene)
2. Posibilidad de añadir materiales de estudio (organizados como libros, notas o simples ficheros)
3. Diferentes tipos de tests y exámenes
4. Laboratorio Virtual con interfaz interactivo (para problemas de electrónica)
5. Caledandario/Planificación del curso
6. Soporte multi-idioma
7. Foros de discusión
8. Wikis
9. Informes de progreso
10. Sistema para implementar *Learning Analytics*
11. Diferentes tipos de evaluación de tareas: evaluación entre pares, auto-evaluación, hetero-evaluación, evaluación automática
12. Sistema de notificación de eventos por correo electrónico
13. Emisión de certificados de completamiento
14. Integración con Google Hangouts
15. Preparado desde el principio para ser escalable

2.3.2. Socios fundadores e instituciones participantes

A junio de 2014 hay 40 socios fundadores y 13 instituciones que participan en el proyecto [28]:

Socios Fundadores

EEUU:

- Massachusetts Institute of Technology
- Berklee College of Music
- California Institute of Technology
- Cornell University
- Davidson College
- Harvard University
- Boston University
- University of Texas System
- University of Washington
- Dartmouth College
- Georgetown University
- Notre Dame
- Rice University
- University of California,
- University of Chicago
- Columbia University
- Wellesley College

Australia:

- Australian National University
- University of Adelaide
- University of Queensland

Canadá:

- McGill University
- University of Toronto

China:

- Tsinghua University
- Peking University

France:

- Sorbonne University

Hong Kong:

- Hong Kong University of Science and Technology
- University of Hong Kong
- The Hong Kong Polytechnic University

Japan:

- Kyoto University
- Tokyo University

Switzerland:

- École Polytechnique Fédérale de Lausanne
- ETH Zurich

Germany:

- Technical University Munich

Netherlands:

- Delft University of Technology
- Wageningen University

Sweden:

- Karolinska Institute

Belgium:

- Université catholique de Louvain

India:

- Birla Institute of Technology and Science
- Indian Institute of Technology, Bombay

South Korea:

- Seoul National University

Instituciones participantes

- Colgate University
- GEMS Education
- Hamilton College
- Inter-American Development Bank
- International Monetary Fund
- Smithsonian Institution
- Learning by Giving Foundation
- Linux Foundation
- OpenCourseWare
- Osaka University
- Universidad Autónoma de Madrid
- Secretariat of Public Education (Mexico)
- **Universidad Carlos III de Madrid**

2.3.3. Presente y futuro de la plataforma

El proyecto se hizo realidad gracias a la inversión total de 60 millones de dólares por parte de sus creadores, sin embargo no se puede predecir si será un modelo sostenible en el tiempo o seguirá la misma suerte que AllLearn o Fathom.com, como se cuenta en el apartado 2.2.3. El responsable del programa, Anant Agarwal, asegura que ni la Universidad de Harvard ni el MIT tienen previsto ingresar dinero a través de este

programa ya que fue creado como un proyecto no lucrativo. Actualmente se están empezando a poner en práctica algunos métodos de financiación, como se adelantó en el apartado 2.2.5, por ejemplo, el cobro en algunos cursos, pocos todavía, de una tasa por homologar los títulos de aprovechamiento y convertirlos en certificados oficiales.

En julio de 2013 se lanzó OpenEdX, la iniciativa de código abierto de edX, para que desarrolladores de todo el mundo puedan utilizar y mejorar la plataforma creando nuevas aplicaciones y añadiendo nuevas funcionalidades. Este es el caso de este proyecto, que surge con la idea de aportar una nueva funcionalidad, la de recomendar recursos a los alumnos en función de su desempeño a lo largo del curso.

Como hemos dicho en el apartado anterior no se puede predecir el futuro de edX, pero hasta el momento no han faltado nuevas ideas, por ejemplo, el 10 de septiembre de 2013, edX anunció su asociación con Google [29] para el desarrollo conjunto de nuevas herramientas de aprendizaje. Como parte de esta colaboración, edX está construyendo MOOC.org, un nuevo sitio web que ayudará a las instituciones educativas, empresas y profesores a crear cursos en línea para una audiencia global. El sitio estará propulsado por edX y construido sobre la infraestructura de Google.

2.3.4. Arquitectura de la plataforma

En este apartado se explica a grandes rasgos la arquitectura de la plataforma edX, se detallará en apartados posteriores.

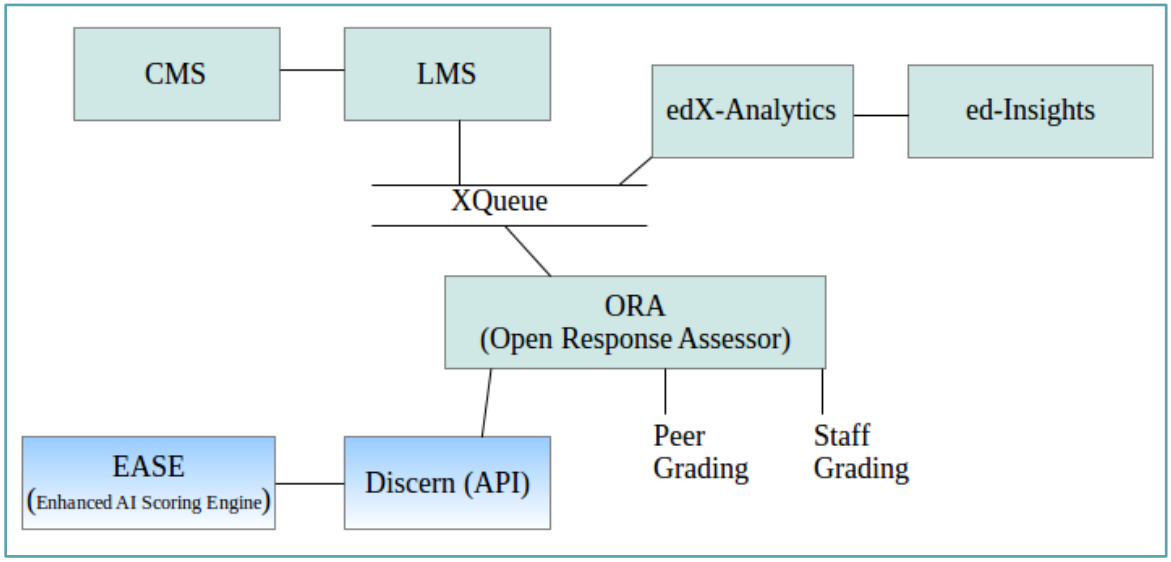


FIGURA 2-12. ARQUITECTURA DE LA PLATAFORMA EDX (SACADA DE [30])

EdX consta de varios componentes, como se muestra en la figura 2-12. Sabemos que una de sus características principales es que debe ser escalable, por lo que se basa en una arquitectura de servicios, una serie de piezas de *software* que se puede ejecutar en equipos independientes y ampliarse si fuera necesario.

Además de los componentes anteriores, edX, utiliza dos sistemas de gestión de bases de datos:

- ***MongoDB***: es un sistema de bases de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, *MongoDB* guarda estructuras de datos en documentos tipo *JSON* con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. En edX almacena el contenido pedagógico, es decir, el contenido de los cursos y los debates o foros de discusión.
- ***SQLite/MySQL***: en entornos localdev se utiliza *SQLite* como sistema de gestión de bases de datos relacional, almacena los datos de registro de los usuarios, las inscripciones a los cursos, los progresos, el estado, etc. En entornos de producción se utiliza *MySQL*.

Dos de los componentes más importantes de la plataforma son el CMS y el LMS, dos aplicaciones de *Django* que funcionan tanto en entornos de producción como de desarrollo [31]:

- **CMS**: es el sistema de gestión de cursos (edX Studio). Es la parte dónde los profesores crean y modifican los cursos. Se comunica con el LMS a través de la base de datos *MongoDB*. Su apariencia se muestra en la figura 2-13.

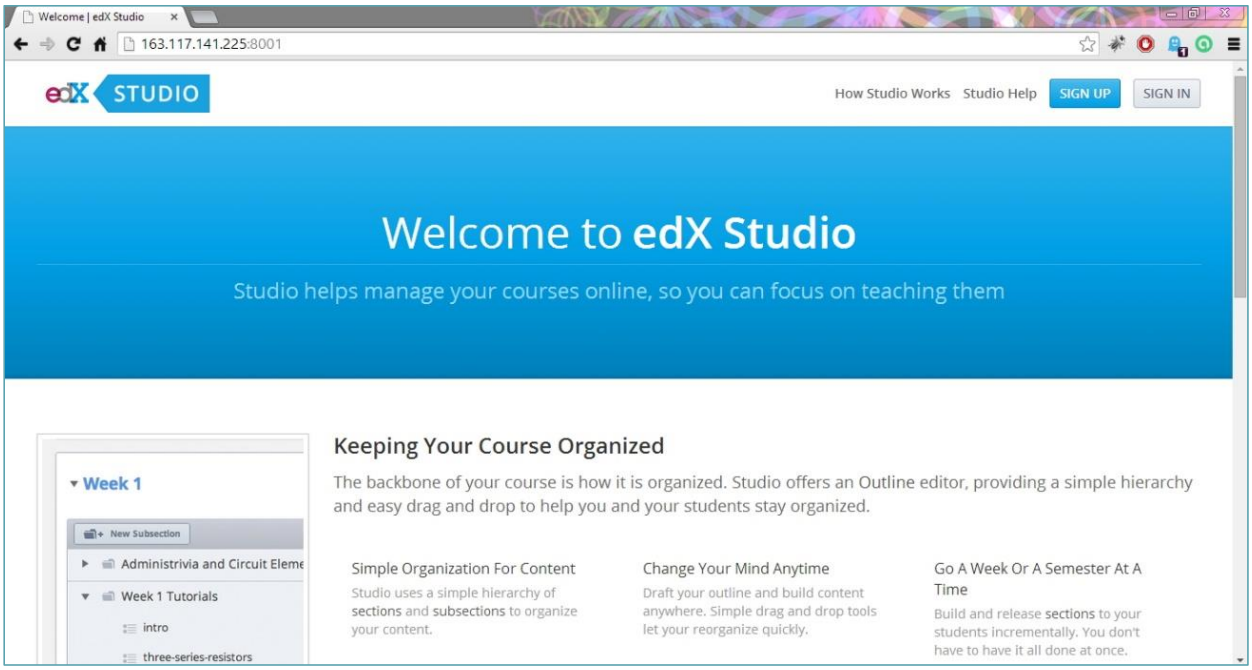


FIGURA 2-13. CAPTURA DEL CMS DE EDX

- **LMS**: es el sistema de gestión de aprendizaje. Es la parte que maneja el estudiante y donde se muestra el contenido (vídeos, problemas, tutoriales, etc). La captura 2-14 es una muestra de la página principal, donde los alumnos se loguean en la plataforma.

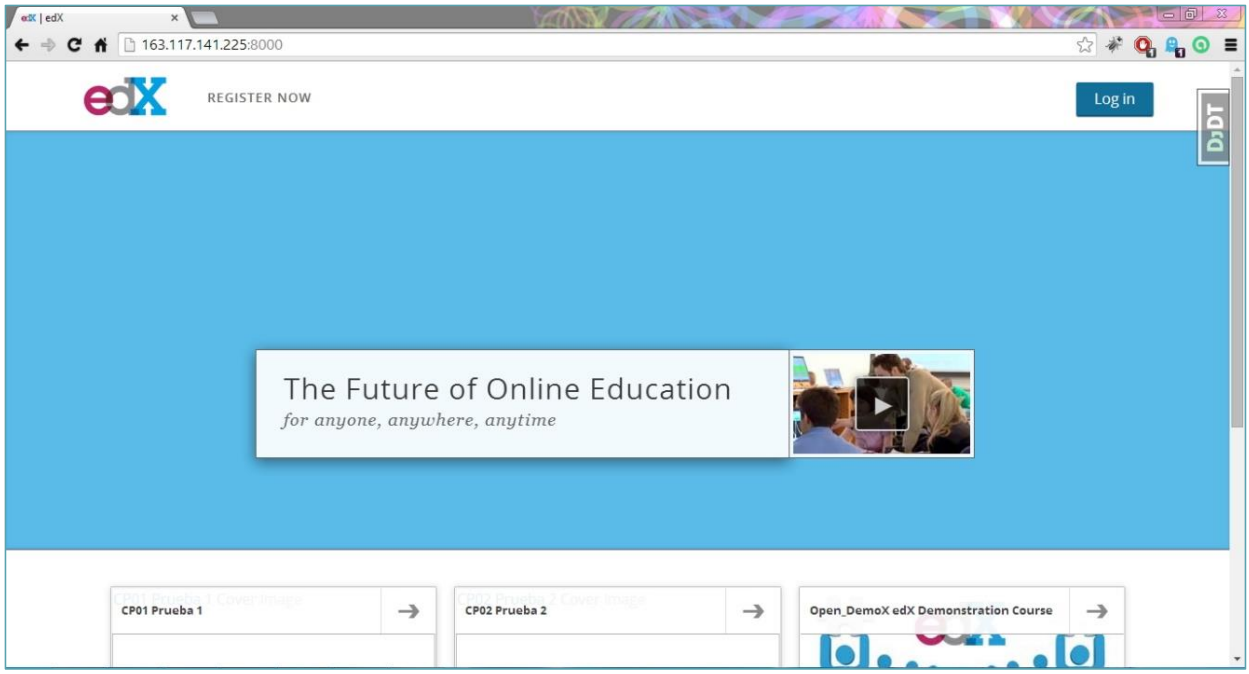


FIGURA 2-14. CAPTURA DEL LMS DE EDX

2.4. Sistemas de recomendación

2.4.1. Definición

Los sistemas, plataformas o motores de recomendación, son un tipo de sistemas de filtrado de información que se encargan de predecir la preferencia del usuario por un elemento [32] o aquellos elementos que pudieran ser mejor para dicho usuario. Una manera de realizar la recomendación es fijarnos en aquellos individuos que tienen gustos similares a los del usuario o elementos con características comunes a otros elementos que el usuario haya comprado, visto o por los que haya mostrado interés en el pasado, y los recomienda.

Debido a sus numerosos campos de aplicación, los sistemas de recomendación, se han hecho muy populares en los últimos años. Pueden recomendar desde películas, música, noticias, artículos de investigación, consultas, etiquetas, y productos en general, hasta restaurantes, productos financieros, seguros de vida, personas o seguidores en Twitter.

De forma generalizada podemos hablar de tres tipos de sistemas de recomendación principales, que serán explicados más en detalle en los siguientes apartados: sistemas de recomendación colaborativos, sistemas de recomendación basados en contenido y sistemas de recomendación híbridos. Sin embargo, también podemos destacar otros sistemas, como los basados en la demografía, que utilizan las características demográficas obtenidas de los datos del perfil del usuario [33]; los sistemas de recomendación basados en utilidad, que requieren información de cómo un elemento satisface las necesidades del usuario [34, 35] y los sistemas de recomendación basados en conocimiento, que parten del conocimiento que da el usuario sobre sus necesidades.

2.4.2. Sistemas de recomendación colaborativos

Los sistemas de recomendación colaborativos, también llamados de filtrado colaborativo o *Collaborative Filtering* (CF, por sus siglas en inglés), se fundamentan en la suposición de que si los usuarios X e Y puntúan n elementos de forma similar o se comportan de forma similar (por ejemplo, comprando, viendo o escuchando), se van a comportar con otros elementos de forma similar [36]. Es decir, hace predicciones sobre los intereses de un usuario mediante la recopilación de las preferencias o gustos de muchos usuarios, convertidos en colaboradores.

Se podría decir que la recomendación se realiza en dos pasos, primero se buscan los usuarios que comparten los mismos patrones de evaluación con el usuario al que se va a recomendar y después se utilizan las evaluaciones de estos usuarios afines para calcular una predicción. Explicado de una manera más formal, podemos imaginarnos un escenario en el que tenemos m usuarios, $\{u_1, u_2, \dots, u_m\}$, y n elementos, $\{i_1, i_2, \dots, i_n\}$, y cada usuario tiene una lista de elementos, Iu_i, puntuados de forma explícita, por ejemplo, mediante una escala del 1 al 5, o inferidos de su comportamiento de forma implícita, por ejemplo viendo qué otros elementos ha comprado o en cuáles ha mostrado interés. Se obtiene así una matriz usuario-elemento de recomendación [37].

Existen dos tipos de algoritmos para estos sistemas:

- **Basados en memoria**: se utilizan todos o una muestra de los datos de las evaluaciones de los usuarios para calcular la similitud entre los mismos. El algoritmo del vecino más cercano es el más frecuente, y tiene lugar en dos pasos: calcular la

similitud o peso, $w_{i,j}$, que representa la distancia o correlación entre dos usuarios o elementos, i y j; y generar una recomendación para el usuario cogiendo la media ponderada de todas las calificaciones de un usuario para un cierto elemento o usuario, o utilizando una media ponderada simple.

En el caso de los algoritmos basados en elementos, para calcular la similitud entre el elemento i y el j, se buscan los usuarios que han calificado los dos elementos y después, mediante una técnica de cálculo de similitudes se determina la similitud $w_{i,j}$ entre ambos elementos. En el caso de los algoritmos basados en usuario, primero se calcula la similitud, $w_{u,v}$, entre los usuarios u y v, los cuales han calificado los mismos elementos [38].

De entre las técnicas de cálculo de similitudes, podemos destacar la similitud basada en la correlación (la Correlación de Pearson y sus variaciones [39], la Correlación por Rangos de Spearman y la Correlación de Kendall [40, 41]), la similitud basada en el coseno entre vectores [42] y la similitud basada en probabilidad [43].

- **Basados en modelos**: el diseño y desarrollo de modelos, como en el caso de los algoritmos de aprendizaje máquina o de minería de datos, permiten al sistema encontrar patrones basándose en unos datos de entrenamiento y, posteriormente, hacer predicciones inteligentes con datos reales. Se utiliza la matriz de calificaciones para crear un modelo a través del cual establecer el conjunto de usuarios similares al usuario activo. En este tipo de algoritmos encontramos clasificadores bayesianos [39], redes neuronales, algoritmos genéticos, modelos de *clustering* [44], modelos basados en regresión [45], modelos basados en procesos de decisión de Markov o modelos basados en técnicas de descomposición matricial, SVD (*Singular Value Decomposition*).

Este tipo de algoritmos tiene una ventaja sobre los basados en memoria, y es que maneja mejor la dispersión de datos y por lo tanto mejora la escalabilidad, sin embargo el costo de construcción del modelo es elevado.

Sin embargo, el aumento de la utilización de este tipo de sistemas ha dado lugar a la aparición de numerosos problemas que antes no se habían tenido en cuenta y que actualmente suponen un gran desafío:

- **Escasez de datos**: en ocasiones la matriz usuario-elemento es muy grande y dispersa y es difícil hacer una recomendación. Cuando un usuario es nuevo se puede dar el problema típico de arranque en frío o *cold start*, en el que el sistema no puede extraer inferencias ya que el usuario aún no ha evaluado el suficiente número de elementos para permitir al sistema recopilar sus preferencias con precisión y poder hacer recomendaciones fiables. Igualmente, cuando se añaden nuevos elementos, tienen que ser valorados por el suficiente número de usuarios antes de poder ser recomendados [46, 47].

- **Escalabilidad**: cuando el número de usuarios y elementos es muy elevado, por ejemplo, del orden de las decenas de millones, muchos algoritmos sufren serios problemas de escalabilidad por lo que inferir recomendaciones es algo prácticamente inviable.

- **Sinonimia**: es la tendencia a dar diferentes nombres a elementos iguales o muy similares. La mayoría de los sistemas de recomendación no tienen la capacidad de descubrir la asociación entre ellos y por tanto, son tratados como si fueran elementos diferentes.

- ***Grey Sheep* (Oveja Gris)**: hace referencia a los usuarios cuyas opiniones no están en claro acuerdo o desacuerdo con ningún grupo de usuarios [48]. En el otro extremo, un usuario cuyos gustos idiosincráticos hagan que la tarea de recomendación sea algo casi imposible, será una *Black Sheep* (Oveja Negra). Debido a que los recomendadores no-electrónicos también tienen problemas recomendando a este tipo de usuarios podemos decir que este es un fallo aceptable [49].

- ***Shilling attacks***: en ocasiones, en sistemas donde todo el mundo puede dar sus opiniones, los usuarios evalúan sus propios artículos de forma positiva y de forma negativa los de sus competidores. Es conveniente tomar precauciones para evitar este tipo de manipulación [50].

- **Diversidad**: los algoritmos no pueden recomendar elementos con escasos datos históricos, por lo que en vez de ayudar a descubrir nuevos productos siempre se tenderá a recomendar los productos más populares. Se entra así en un bucle en el que los nuevos productos quedan apartados debido a sus escasas ventas o calificaciones, fenómeno que desemboca en una disminución de la diversidad [51].

2.4.3. Sistemas de recomendación basados en contenido

Los sistemas de recomendación basados en contenido analizan el contenido de los textos, como documentos, *URLs*, noticias, *weblogs*, descripciones de artículos y los perfiles de los usuarios para conocer sus preferencias y necesidades [52]. Posteriormente, el sistema, de forma automática, trata de recomendar elementos que el usuario ya había adquirido o consumido en el pasado basándose en su perfil.

De una forma más concreta, cuando un usuario encuentra interesante un documento o un elemento, el sistema puede facilitarle contenidos similares dependiendo de los pesos que se asignan a las palabras que describen el contenido del documento o a las características del elemento dentro del perfil del usuario, es decir, se accede a su perfil de usuario para recomendarle otros elementos que pueden ser de su agrado. A este proceso se le conoce como *feedback* por relevancia.

Este tipo de sistemas también tienen sus inconvenientes, el más importante es que solo ciertos tipos de contenidos pueden ser analizados, principalmente texto. Para otras fuentes multimedia, como video o audio, la predicción es mucho más complicada. Por otro lado, tienen el problema de la puesta en marcha, debe existir suficiente información para obtener una recomendación fiable y no todos los usuarios están dispuestos a puntuar las informaciones que les son ofrecidas por lo que no siempre se obtiene un *feedback* adecuado. Es también importante el problema de exceso de especialización de ciertos usuarios, es decir, solo se recomiendan artículos con una valoración elevada en el perfil del usuario o en su historial de calificación, por lo que el usuario vería restringidas sus recomendaciones a ese tipo de elementos.

2.4.4. Sistemas de recomendación híbridos

Los sistemas de recomendación híbridos combinan técnicas de filtrado colaborativo con otras técnicas de recomendación, generalmente con sistemas basados en contenido, con el objetivo de minimizar debilidades.

Más en detalle, podemos encontrar sistemas híbridos en los que se han añadido características de los sistemas de filtrado basados en contenido a un modelo de filtrado colaborativo, otros en los que se han añadido características de los sistemas de filtrado colaborativo a un modelo basado en contenido, una combinación de ambos o sistemas como combinación de diferentes algoritmos de filtrado colaborativo [53, 54].

2.5. Sistemas de recomendación en TEL (Technology Enhanced Learning)

Según lo explicado en el apartado 2.1 del presente documento podemos decir que el e-learning es un tipo de aprendizaje potenciado por la tecnología (*Technology Enhanced Learning*, TEL), o incluso, que ambos términos son sinónimos. Es una modalidad educativa basada en la utilización de Internet como plataforma de enseñanza y aprendizaje y, por lo tanto, los recursos disponibles en la red, al igual que los productos, también son susceptibles de ser calificados por los usuarios y recomendados.

En el caso de los sistemas de recomendación aplicados al e-learning no solo se recomiendan contenidos digitales almacenados en repositorios como MERLOT, OER Commons o European Schoolnet's Learning Resource Exchange, si no que se va un paso más allá, se generan caminos de aprendizaje por los que se guía al estudiante recomendándole los recursos más apropiados para él con el fin de mejorar su experiencia o se recomiendan otros usuarios con los que llevar a cabo actividades de aprendizaje colaborativo [55].

Herlocker et al. (2004) [40], hace un análisis de las tareas llevadas a cabo por los usuarios soportadas por los sistemas de recomendación actuales. En la primera parte de la tabla 2-4 se incluye una descripción de estas tareas y ejemplos de recomendación tanto para los sistemas de recomendación de carácter general como para los sistemas de recomendación en TEL. Observamos que para cubrir algunas de las tareas, en el caso de los sistemas de recomendación en TEL, es necesario algún requisito adicional. La segunda parte de la tabla añade tareas específicas más orientadas a las necesidades de recomendación en TEL.

TABLA 2-4. TAREAS SOPORTADAS POR LOS SISTEMAS DE RECOMENDACIÓN ACTUALES Y REQUISITOS PARA LOS SISTEMAS DE RECOMENDACIÓN EN TEL (SACADA DE [40])

Tasks	Description	Generic recommender	TEL recommenders	New requirements
Existing User Tasks supported by Recommender Systems				
Annotation in context	Recommendation while user carries out other tasks	Eg. Predictig how relevant the links are within a web page	Eg. Predicting relevance/usefulness of ítems in the reading list of a Moodle course on a Learning Network	Explore attributes for representing relevance/usefulness in a learning context
Find good ítems	Recommendation of suggested ítems	Eg. Receiving list of web pages to visit	Eg. Receving a selected list of online educational resources around a topic	None
Find all good ítems	Recommendatino of all relevant ítems	Eg. Receiving a complete list of references on a topic	Eg. Suggesting a complete list of scientific literature or blog postings around a topic	None
Recommend sequence	Recommendation of a sequence of ítems	Eg. Receive a proposed sequence of songs	Eg. Receiving a proposed sequence through resources to achieve a particular learning goal	Explore formal and informal attributes for representing relevancy to a particular learning goal
.Just browsing	Recommendations out of the box while user is browsing	Eg. People that bought this have also bought that	Eg. Receiving recommendations for new courses on the university site or getting suggestions for additional blog postings in a Learning Network	Explore formal and informal attributes for representing relevance/usefulness in a learning context
Find credible recommender	Recommendations during initial exploration/testing phase of a system	Eg. Movies that you will definitely like	Eg. Restricting initial recommendations to ones with high confidence credibility	Explore criteria for measuring confidence and credibility in formal and informal learning

TEL User Tasks that could be supported by Recommender Systems				
Find novel resources	Recommendations of particularly new or novel ítems	Eg. Receiving recommendations about latest additions or particularly controversial ítems	Eg. Receiving very new and/or controversial resources on covered topics	Explore recommendation techniques that select ítems beyond the similarity
Find peers	Recommendation of other people with relevant interests	Eg. Being suggested profiles of users with similar interests	Eg. Being suggested students in the same class or a peer-student in a Learning Network	Explore attributes for measuring the similarity with other people
Find good pathways	Recommendation of alternative learning paths through learning resources	Eg. Receive alternative sequences of similar songs	Eg. Receiving a list of alternative learning paths over the same resources to achieve a learning goal	Explore criteria for the construction and suggestion of alternative (but similar) sequences

Sin embargo, la afirmación *one-size-fits-all* no puede ser aplicada en este contexto. Existen muchas otras variables que los sistemas de recomendación en TEL deben considerar a la hora de satisfacer las necesidades de los usuarios.

Para abordar esta cuestión de no-heterogeneidad entre los usuarios aparecen los sistemas *Adaptive Educational Hypermedia* (AEH), según Brusilovsky (2001) [56], una de las primeras aplicaciones de los sistemas adaptativos. Estos sistemas están enfocados a entornos educativos formales, como las universidades, donde los planes de estudio están predefinidos y bien estructurados, y se puede ofrecer recomendaciones personalizadas haciendo uso de tecnologías como los metadatos y las ontologías. En el caso de entornos educativos informales, como pueden ser las *Learning Networks*, donde existe un número ilimitado de recursos que no pueden ser estructurados, son más apropiados los sistemas de filtrado colaborativo, explicados en el apartado 2.4.2 del presente documento.

Actualmente existe un gran número de sistemas de recomendación en TEL basados en filtrado colaborativo. En la siguiente tabla se enumeran los sistemas más importantes y su estado actual:

TABLA 2-5. EJEMPLOS DE SISTEMAS DE RECOMENDACIÓN EN TEL (SACADA DE [55])

Sytem	Status	Evaluator focus	Evaluation roles
Altered Vista [57, 58, 59, 60]	Full system	Interface, Algorithm, System usage	Human users
RACOFI [61, 62]	Prototype	Algorithm	System designers
QSAI [63, 64]	Full system	-	-
CYCLADES [65]	Full system	Algorithm	System designers
CoFind [66]	Prototype	System usage	Human users
Learning object sequencing [67]	Prototype	System usage	Human users
Evolving e-learning system [68, 69, 70, 71, 72]	Full system	Algorithm, System usage	Simulated users, Human users
ISIS - Hybrid Personalised Recommender System [73]	Prototype	Algorithm, System usage	Human users
Multi-Attribute Recommendation Service [74]	Prototype	Algorithm	System designers
Learning Object Recommendation Model [75]	Design	-	-
RecoSearch [76]	Design	-	-
Simulation environment [77]	Full system	Algorithm	Simulated users
ReMashed [78, 79]	Full system	Algorithm, System usage	Human users
CourseRank [80, 81]	Full system	System usage	Human users
CBR Recommender Interface [82]	Prototype	-	-
APOSDLE Recommendation Service [83]	Prototype	-	-
A2M Recommending System [84]	Prototype	-	-
Moodle Recommender System [85]	Prototype	Algorithm, System usage	Human users
LRLS [86]	Prototype	System usage, Learner performance	Human users
RPL recommender [87]	Prototype	System usage	System designers, Human users

Como podemos observar la mayoría de los sistemas están en desarrollo o no han sido evaluados ni siquiera de forma experimental, prueba indispensable antes de la implementación del sistema en un entorno real [88].

2.6. Evaluación de los sistemas de recomendación

Según [89] las métricas de evaluación de los sistemas de recomendación se clasifican en: precisión de predicción, como el error absoluto medio (MAE), el error absoluto medio normalizado (NMAE) y el error cuadrático medio (RMSE), precisión de clasificación como la Característica Operativa del Receptor o Curva ROC, y precisión de rango como la correlación de Pearson, la Tau de Kendall o la métrica de rendimiento basada en la distancia normalizada (NDPM) [90].

La medida más utilizada es el error absoluto medio, MAE, que calcula el promedio de la diferencia absoluta entre las predicciones y las calificaciones correctas. Cuanto menor sea el error mejor será la predicción:

$$MAE = \frac{\sum_{\{i,j\}} |p_{i,j} - r_{i,j}|}{n},$$

- $p_{i,j}$: predicción para el usuario j sobre el artículo i
- $r_{i,j}$: calificación correcta para el usuario j sobre el artículo i
- n: número total de calificaciones de todos los usuarios

Como solución a la variedad de escalas de calificación podemos normalizar el error absoluto medio y expresarlo como porcentajes:

$$NMAE = \frac{MAE}{r_{max} - r_{min}},$$

- r_{max}: límite superior de la escala de calificaciones
- r_{min}: límite inferior de la escala de calificaciones

Otra medida, muy popular actualmente ya que es la utilizada por Netflix[10] para la recomendación de películas, es el error cuadrático medio:

$$RMSE = \sqrt{\frac{1}{n}\sum_{\{i,j\}}\left(p_{i,j} - r_{i,j}\right)^2},$$

- n: número total de calificaciones de todos los usuarios
- $p_{i,j}$: predicción para el usuario j sobre el artículo i
- $r_{i,j}$: calificación correcta para el usuario j sobre el artículo i

Para calcular la calidad de una clasificación se utiliza el área bajo la curva ROC (AUC), que es la representación en dos dimensiones de la Tasa de Falsos Positivos (eje X) y la Tasa de Positivos Verdaderos (eje Y). Cuanto mayor sea el AUC mejor será la clasificación:

$$AUC = \frac{S_0 - n_0(n_o+1)/2}{n_0 n_1},$$

- n_0 y n_1 son el número de ejemplo negativos y positivos respectivamente
- $S_0 = \sum_{r_i}$ donde r_i es la calificación del $i$ésimo ejemplo positivo en la lista de calificaciones

Sin embargo, la medida de la calidad de un sistema de recomendación en TEL es uno de los principales retos de investigación ya que no se limita a cálculos técnicos, sino que es necesario tener en cuenta el contexto en el que se lleva a cabo el proceso de enseñanza-aprendizaje. La evaluación de estos sistemas es mucho más compleja que la de los sistemas de recomendación generales, ya que, al ser sistemas adaptativos, intervienen muchas más variables y hay que elegir de forma adecuada los criterios de evaluación.

Hasta la fecha se han realizado numerosos estudios acerca de los pasos a seguir para la evaluación de estos sistemas. Todos ellos coinciden en separar este proceso en módulos o capas en los que se identifican los criterios a tener en cuenta a la hora de

[10] http://www.netflixprize.com/

evaluar un sistema adaptativo, así como los métodos y herramientas a utilizar [91]. Se debe estudiar la eficacia, la eficiencia, la satisfacción y la tasa de abandono, o, de forma más general, aplicando en modelo de Kirckpatrick (1959): la reacción de los usuarios ('*Did I enjoy the recommendations I receive?*'), el aprendizaje ('*Did I learn what I needed to and get some new ideas, with the help of the recommender?*'), el comportamiento ('*Will I use the new information and ideas I was recommended?*') y los resultados ('*Do the ideas and information I was recommended improve my effectiveness and results?*

3. Herramientas y tecnologías aplicadas

3.1. Django

3.1.1. Visión general

FIGURA 3-1. LOGO DE DJANGO

Django es un *framework* de desarrollo web de código abierto escrito en Python que utiliza un patrón de arquitectura conocido como *Model-Template-View* (MTV), modificación del Modelo-Vista-Controlador (MVC) [92].

Fue desarrollado por Adrian Holovaty, Simon Willison, Jacob Kaplan-Moss y Wilson Milner como herramienta para la administración de tres sitios de publicación de noticias: The Lawrence Journal-World, Lawrence.com y KUsports.com. Se nombró así en honor al músico francés Django Reinhardt.

Su diseño está orientado a la optimización del tiempo de desarrollo, basándose en la reutilización y en el principio *No Te Repitas* (DRY, por sus siglas en inglés). Tiene una aplicación incorporada con una interfaz para administrar tanto los contenidos como los usuarios y los grupos de usuarios. En la captura de la figura 3-2 se muestra la pantalla de acceso al administrador de *Django* y en la figura 3-3 una captura de la página principal del mismo, donde podemos ver los distintos modelos de objetos, acceder a ellos y modificarlos.

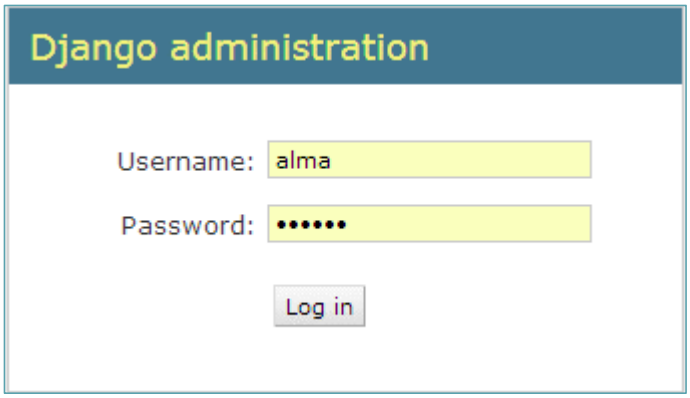


FIGURA 3-2. ACCESO AL ADMINISTRADOR DE *DJANGO*

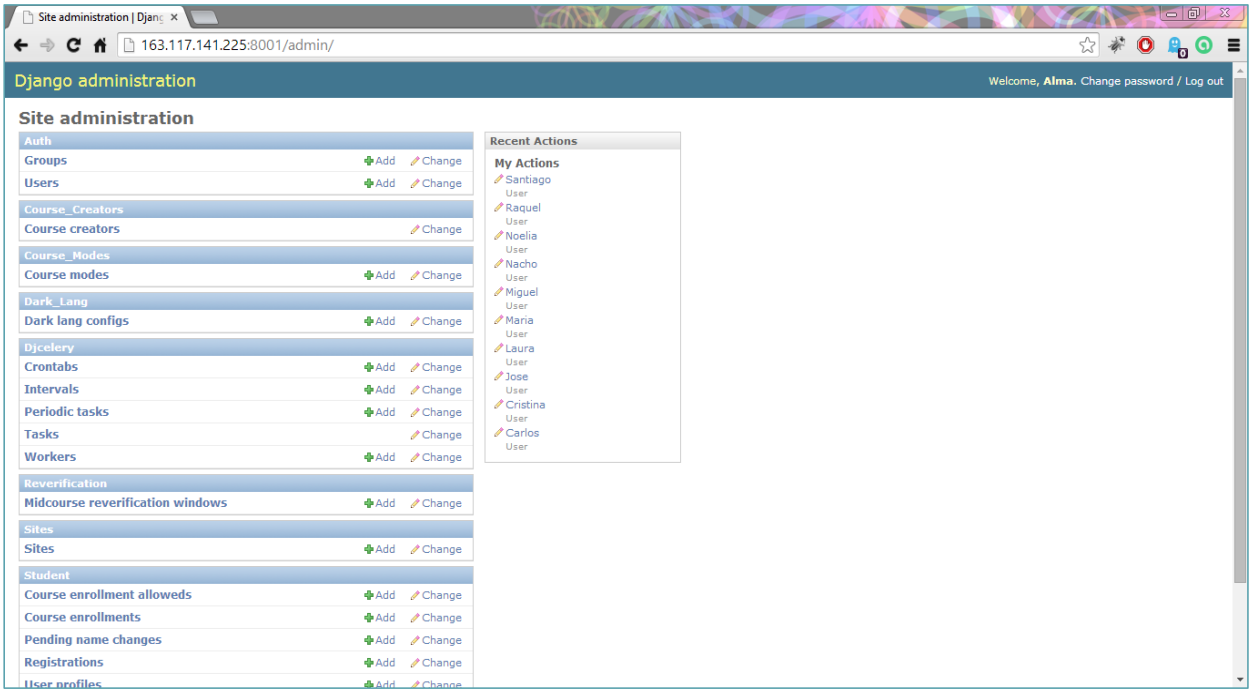


FIGURA 3-3. SITIO DE ADMINISTRACIÓN DE *DJANGO*

Por lo tanto, podemos decir, que *Django* tiene dos partes. En la plataforma edX, para la que se ha realizado este proyecto, son las siguientes:

- Un sitio público, la página web de edX, donde los alumnos se registran y acceden a los cursos ofertados. Pueden apuntarse a los cursos, acceder a los contenidos, realizar los ejercicios, ver vídeos, interactuar entre ellos, etc.

- Un sitio de administración, mediante el cual el administrador o administradores se encargan de gestionar toda la información referente a los usuarios o grupos de usuarios.

Actualmente, *Django* está presente en numerosos sitios web, algunos de los más conocidos son: Pinterest, Instagram, BitBucket, Mozilla Support, The Onion, Disqus, Mahalo, Fluendo y edX (en el que nos centramos en este proyecto); y muchos sitios de noticias como The Guardian, The New York Times, The Washington Post o el mencionado anteriormente Lawrence.com.

3.1.2. Arquitectura

En la imagen 3-4 se representa de forma general la arquitectura de *Django*. Como ya dijimos en el apartado anterior, *Django*, sigue un patrón MTV, a continuación explicamos en detalle la función de cada componente [93]:

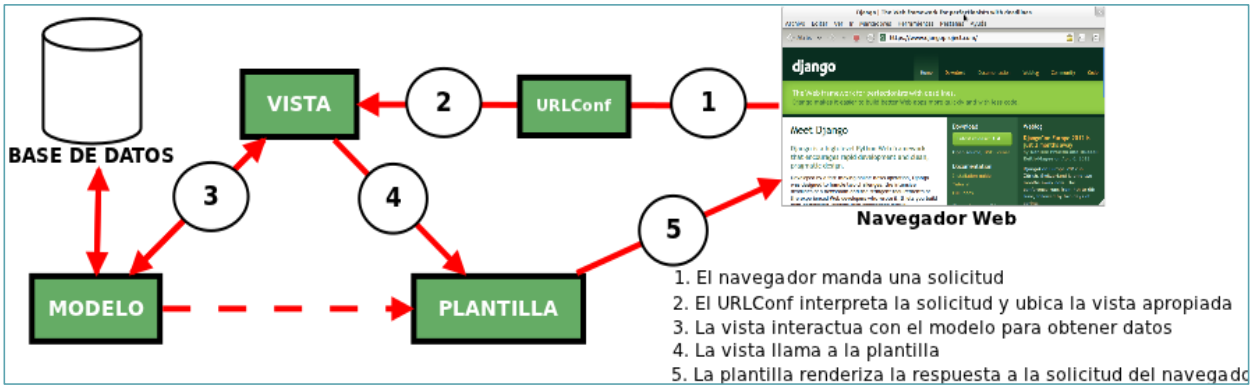


FIGURA 3-4. ARQUITECTURA DE *DJANGO*. (SACADA DE [93])

- **Modelo (*Model*):** es una capa de abstracción para la estructuración y manipulación de los datos, o, de forma más sencilla, se encarga de definir los datos almacenados en forma de clases Python. Los tipos de datos se almacenan en variables con varios parámetros y pueden tener métodos para controlarlos a nivel de fila, es decir, métodos que actúan en una misma instancia de la clase. En la imagen 3-5 podemos ver un ejemplo de un modelo de *Django*, codificado en la clase de Python `UserPreference`.

Para este proyecto solo ha sido necesaria la creación de un modelo de *Django*, que se corresponde con una tabla ('recommender_student') de la base de datos en *MySQL*. Cada instancia de esta clase será una fila de la tabla donde se almacena cada alumno ('user_id') y sus problemas ('module_id') recomendados para cada curso ('course_id').

```
from django.contrib.auth.models import User
from django.db import models


class UserPreference(models.Model):
    """A user's preference, stored as generic text to be processed by client"""
    user = models.ForeignKey(User, db_index=True, related_name="+")
    key = models.CharField(max_length=255, db_index=True)
    value = models.TextField()

    class Meta:
        unique_together = ("user", "key")
```

FIGURA 3-5. EJEMPLO DE MODELO DE *DJANGO*

- **Vista (*View*)**: formada por funciones en Python, se encarga de encapsular la lógica responsable de procesar las solicitudes del usuario (*Request*) y las respuestas (*Response*). Determina qué datos serán visualizados (no el estilo ni la forma en que serán visualizados, de eso se encarga la plantilla). El ORM (del inglés *Object-Relational Mapping*) hace que no sea necesario utilizar código *SQL* para realizar las consultas. En la captura de la imagen 3-6 vemos un ejemplo de una clase *View* de *Django*.

```
from django.conf import settings
from edxmako.shortcuts import render_to_response

from multicourse import multicourse_settings


def edxhome(request):
    ''' Home page (link from main header). List of courses.  '''
    if settings.DEBUG:
        print "[djangoapps.multicourse.edxhome] EDX_ROOT_URL = " + settings.EDX_ROOT_URL
    if settings.ENABLE_MULTICOURSE:
        context = {'courseinfo': multicourse_settings.COURSE_SETTINGS}
        return render_to_response("edXhome.html", context)
    return info(request)
```

FIGURA 3-6. EJEMPLO DE *VIEW* DE *DJANGO*

- **Plantilla (*Template*)**: página *HTML* con etiquetas y filtros exclusivos de *Django*. Recibe los datos de la vista y los presenta al usuario en el navegador de una forma sencilla.

```
<%! from django.utils.translation import ugettext as _ %>
<%inherit file="/main.html" />
<%! from django.core.urlresolvers import reverse %>
<%namespace name='static' file='/static_content.html'/>

<%include file="/courseware/course_navigation.html" args="active_page=''" />

<section class="container">
<div class="gradebook-summary-wrapper">
  <section class="gradebook-summary-content">
    <h1>${_("Grade summary")}</h1>

    <p>${_("Not implemented yet")}</p>

  </section>
</div>
</section>
```

FIGURA 3-7. EJEMPLO DE *TEMPLATE* DE *DJANGO*

Como podemos observar en la imagen 3-8 hay un módulo adicional, URLConf, que se encarga de mapear las direcciones *URL* con las *Views* de la aplicación, es decir, lee la *URL* solicitada por el usuario, encuentra la vista apropiada y pasa las variables que la vista necesite.

```
from django.conf.urls import patterns, url
from django.conf import settings

urlpatterns = patterns('shoppingcart.views',  # nopep8
    url(r'^postpay_callback/$', 'postpay_callback'),
    url(r'^receipt/(?P<ordernum>[0-9]*)/$', 'show_receipt'),
    url(r'^csv_report/$', 'csv_report', name='payment_csv_report'),
)
```

FIGURA 3-8. EJEMPLO DE URLCONF

3.1.3. Proceso de un Request

A continuación se explica a grandes rasgos todo el proceso que se lleva a cabo desde que se recibe una solicitud de un usuario (*Request*) hasta que se genera una respuesta (*Response*):

1. Django recibe un *Request* que convierte en un objeto *HttpRequest* que sirve como abstracción para trabajar sobre diferentes servidores.
2. Se resuelve la *URL*, es decir, se selecciona la función de la *View* que participará en la creación de la respuesta:
    1. *Django* determina qué módulo raíz de URLConf utilizar. Normalmente, se indica en el fichero 'settings' y es el valor de la variable `ROOT_URLCONF` pero si el objeto `HttpRequest` tiene un atributo `urlconf` se utiliza en lugar del valor de `ROOT_URLCONF`.
    2. *Django* carga el módulo anterior y busca la variable `urlpatterns` (una lista de instancias del tipo `django.conf.urls.url()`).
    3. *Django* recorre la lista de patrones *URL* hasta que encuentra uno coincidente con la *URL* requerida. Si no hay patrones coincidentes o si se produce alguna excepción durante el proceso, *Django* llama la *View* apropiada que maneja el error.
    4. *Django* invoca a la función de la *View* con el objeto `Request` como primer parámetro y le pasa el resto de argumentos necesarios.
3. La *View* interactúa con el *Model* para obtener datos. A partir de aquí es cuando se realiza el trabajo pesado, como el acceso a las bases de datos.
4. La *View* llama al *Template.*
5. El *Template* renderiza la respuesta a la solicitud del navegador generando el *HTML* que se mostrará al usuario en pantalla.

3.1.4. Ficheros por defecto y modificaciones

Cuando se crea un proyecto de *Django* se generan una serie de ficheros de forma automática:

- **__init__.py**: es un fichero vacío que le dice a Python que el directorio en el que se encuentra debe considerarse como un paquete de Python.
- **manage.py**: contiene una herramienta de línea de comandos que permite interactuar con el proyecto.
- **settings.py**: contiene la configuración del proyecto.
- **urls.py**: contiene todas las URLs del proyecto. Es manejado por el módulo URLConf.
- **wsgi.py**: es el punto de entrada para los servidores web compatibles con WSGI. Especifica la interfaz mediante la cual se comunican el servidor y la aplicación.

Lo mismo ocurre cuando se crea una aplicación para un proyecto de *Django*:

- **__init__.py**: al igual que el del proyecto, es un fichero vacío que le dice a Python que el directorio en el que se encuentra debe considerarse como un paquete de Python.
- **models.py**: donde se declaran las clases del modelo. Cada una se corresponde con una tabla en la base de datos.
- **views.py**: donde se declaran las funciones de la vista.
- **tests.py**: donde se declaran las pruebas necesarias para la aplicación.

En este trabajo se desarrolla una aplicación para el proyecto de *Django* edX, por lo tanto ha sido necesario integrar nuestro código con el del proyecto ya existente. Para ello se ha realizado un estudio exhaustivo de las distintas carpetas y ficheros y se han tenido que realizar varias modificaciones en el código.

Una vez que se han generado los ficheros anteriores los cambios realizados en el código, como la modificación de algunos parámetros de configuración, para satisfacer las necesidades tanto del proyecto edX como de nuestra aplicación 'recommender' han sido los siguientes:

- La base de datos que se utiliza en *Django* por defecto es *SQLite*. En edX se utiliza *MySQL* y se configura en un fichero llamado `/edx/app/edxapp/edx-platform/lms/envs/dev.py` (ver figura 3-9).

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': ENV_ROOT / "db" / "edx.db",
    }
}
```

FIGURA 3-9. CONFIGURACIÓN DE LA BASE DE DATOS DE EDX

- En edX los parámetros de configuración se encuentran en un fichero llamado `/edx/app/edxapp/edx-platform/lms/envs/common.py`, donde encontramos la configuración de la zona horaria y del idioma (ver figura 3-10).

```
# Locale/Internationalization
TIME_ZONE = 'America/New_York'  # http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
LANGUAGE_CODE = 'en'  # http://www.i18nguy.com/unicode/language-identifiers.html
```

FIGURA 3-10. CONFIGURACIÓN DE LA ZONA HORARIA Y DEL IDIOMA DE EDX

- En ese mismo fichero se encuentra el parámetro `INSTALLED_APPS`, una lista de aplicaciones donde se incluyen aquellas que han sido activadas para esta instancia de *Django*. Aquí, como vemos en la figura 3-11, es donde se ha añadido la aplicación 'recommender' creada para este proyecto.

```
INSTALLED_APPS = (
    # Standard ones that are always installed...
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.humanize',
    'django.contrib.messages',
    'django.contrib.sessions',
    'django.contrib.sites',
    'djcelery',
    'south',

    # Database-backed configuration
    'config_models',

    # Monitor the status of services
    'service_status',

    # For asset pipelining
    'edxmako',
    'pipeline',
    'staticfiles',
    'static_replace',

    # Our courseware
    'circuit',
    'courseware',
    'lms.lib.perfstats',
    'student',
    'static_template_view',
    'staticbook',
    'track',
    'eventtracking.django',
    'util',
    'certificates',
    'dashboard',
    'instructor',
    'instructor_task',
    'open_ended_grading',
    'psychometrics',
    'licenses',
    'course_groups',
    'bulk_email',
    'recommender',

    # External auth (OpenID, shib)
    'external_auth',
    'django_openid_auth',

    # For the wiki
    'wiki',  # The new django-wiki from benjaoming
    'django_notify',
    'course_wiki',  # Our customizations
    'mptt',
    'sekizai',
    #'wiki.plugins.attachments',
    'wiki.plugins.links',
    'wiki.plugins.notifications',
    'course_wiki.plugins.markdownedx',

    # Foldit integration
    'foldit',

    # For A/B testing
    'waffle',

    # For testing
    'django.contrib.admin',  # only used in DEBUG mode
    'django_nose',
    'debug',

    # Discussion forums
    'django_comment_client',
    'django_comment_common',
    'notes',

    # Splash screen
    'splash',

    # Monitoring
    'datadog',

    # User API
    'rest_framework',
    'user_api',

    # Shopping cart
    'shoppingcart',

    # Notification preferences setting
    'notification_prefs',

    # Different Course Modes
    'course_modes',

    # Student Identity Verification
    'verify_student',

    # Dark-launching languages
    'dark_lang',

    # Microsite configuration
    'microsite_configuration',

    # Student Identity Reverification
    'reverification',
)
```

FIGURA 3-11. MODIFICACIÓN DE INSTALLED_APPS

Una vez que hemos añadido nuestra app a la lista de aplicaciones debemos comunicar a *Django* que hemos realizado ciertos cambios, para ello utilizamos los siguientes comandos:

- `edxapp@precise64:~/edx-platform$./manage.py lms schemamigration recommender --initial --settings=devstack`: para almacenar los cambios como una `migration` utilizando la configuración de la pila de desarrollo o developer stack. En este caso el cambio es la integración de nuestra aplicación 'recommender'. Vemos un ejemplo de la salida obtenida en la captura 3-12.

```
edxapp@precise64:~/edx-platform$ ./manage.py lms migrate recommender --settings=
devstack
Running migrations for recommender:
 - Migrating forwards to 0001_initial.
 > recommender:0001_initial
 - Loading initial data for recommender.
Installed 0 object(s) from 0 fixture(s)
edxapp@precise64:~/edx-platform$
```

FIGURA 3-12. SALIDA AL APLICAR EL COMANDO SCHEMAMIGRATION

- `edxapp@precise64:~/edx-platform$./manage.py lms migrate recommender --settings=devstack`: para aplicar la `migration` obtenida en el paso anterior, es decir, para crear el nuevo modelo (recommender.Student) en la base de datos. En la imagen 3-13 vemos en mensaje de salida obtenido al ejecutar el comando.

```
edxapp@precise64:~/edx-platform$ ./manage.py lms schemamigration recommender --i
nitial --settings=devstack
 + Added model recommender.Student
Created 0001_initial.py. You can now apply this migration with: ./manage.py migr
ate recommender
edxapp@precise64:~/edx-platform$
```

FIGURA 3-13. SALIDA AL APLICAR EL COMANDO MIGRATION

- Otra modificación de código realizada es, en el fichero `/edx/app/edxapp/edx-platform/lms/urls.py`, añadir la *URL* que apunta a nuestra aplicación, como vemos en la imagen 3-14.

```
urlpatterns = ('',  # nopep8
    # certificate view

    url(r'^update_certificate$', 'certificates.views.update_certificate'),
    url(r'^$', 'branding.views.index', name="root"),   # Main marketing page, or redirect to courseware
    url(r'^dashboard$', 'student.views.dashboard', name="dashboard"),
    url(r'^token$', 'student.views.token', name="token"),
    url(r'^login$', 'student.views.signin_user', name="signin_user"),
    url(r'^register$', 'student.views.register_user', name="register_user"),

    url(r'^admin_dashboard$', 'dashboard.views.dashboard'),

    url(r'^change_email$', 'student.views.change_email_request', name="change_email"),
    url(r'^email_confirm/(?P<key>[^/]*)$', 'student.views.confirm_email_change'),
    url(r'^change_name$', 'student.views.change_name_request', name="change_name"),
    url(r'^accept_name_change$', 'student.views.accept_name_change'),
    url(r'^reject_name_change$', 'student.views.reject_name_change'),
    url(r'^pending_name_changes$', 'student.views.pending_name_changes'),
    url(r'^event$', 'track.views.user_track'),
    url(r'^t/(?P<template>[^/]*)$', 'static_template_view.views.index'),

    url(r'^accounts/login$', 'student.views.accounts_login', name="accounts_login"),
    url(r'^accounts/manage_user_standing', 'student.views.manage_user_standing',
        name='manage_user_standing'),
    url(r'^accounts/disable_account_ajax$', 'student.views.disable_account_ajax',
        name="disable_account_ajax"),

    url(r'^login_ajax$', 'student.views.login_user', name="login"),
    url(r'^login_ajax/(?P<error>[^/]*)$', 'student.views.login_user'),
    url(r'^logout$', 'student.views.logout_user', name='logout'),
    url(r'^create_account$', 'student.views.create_account', name='create_account'),
    url(r'^activate/(?P<key>[^/]*)$', 'student.views.activate_account', name="activate"),

    url(r'^password_reset/$', 'student.views.password_reset', name='password_reset'),
    ## Obsolete Django views for password resets
    ## TODO: Replace with Mako-ized views
    url(r'^password_change/$', django.contrib.auth.views.password_change,
        name='auth_password_change'),
    url(r'^password_change_done/$', django.contrib.auth.views.password_change_done,
        name='auth_password_change_done'),
    url(r'^password_reset_confirm/(?P<uidb36>[0-9A-Za-z]+)-(?P<token>.+)/$',
        'student.views.password_reset_confirm_wrapper',
        name='auth_password_reset_confirm'),
    url(r'^password_reset_complete/$', django.contrib.auth.views.password_reset_complete,
        name='auth_password_reset_complete'),
    url(r'^password_reset_done/$', django.contrib.auth.views.password_reset_done,
        name='auth_password_reset_done'),

    url(r'^heartbeat$', include('heartbeat.urls')),

    url(r'^user_api/', include('user_api.urls')),

    url(r'^', include('waffle.urls')),

    url(r'^recommender/', include('recommender.urls'))
)
```

FIGURA 3-14. MODIFICACIÓN DEL FICHERO URLS.PY

3.2. Vagrant

FIGURA 3-15. LOGO DE VAGRANT

Vagrant es una herramienta de código abierto desarrollada en Ruby por Mitchell Hashimoto y John Bender que nos permite crear y gestionar entornos de desarrollo virtuales fácilmente configurables [94]. Puede gestionar máquinas virtuales alojadas virtualizadores como Oracle VirtualBox, VMWare, AWS, etc.

Una máquina virtual es una aplicación de software que simula el funcionamiento de una máquina real, con un disco virtual, una memoria y una CPU. La máquina en la que se ejecuta el virtualizador es el sistema Host y la máquina virtual que se ejecuta es el sistema Visitante, cuyo sistema operativo se ejecuta en el hardware real y todos los recursos del Host son utilizados por el virtualizador.

La función de *Vagrant* es la gestión de las máquinas virtuales alojadas en los virtualizadores. Cada instancia de máquina virtual tiene un archivo denominado 'Vagrantfile' en el que se describe la configuración de la máquina. En nuestro caso se ha creado en VirtualBox un entorno virtual de desarrollo para la plataforma edX (edX Developer Stack o Devstack) con la misma configuración y el mismo software que la máquina utilizada por los desarrolladores. En la captura de la imagen 3-16 encontramos los parámetros de configuración:

```
Vagrant.configure("2") do |config|

  # Creates an edX devstack VM from an official release
  config.vm.box     = "gugelhupf-devstack"
  config.vm.box_url = "http://files.edx.org/vagrant-images/20140210-gugelhupf-devstack.box"

  config.vm.network :private_network, ip: "192.168.83.10"
  config.vm.network :forwarded_port, guest: 8000, host: 8000
  config.vm.network :forwarded_port, guest: 8001, host: 8001
  config.vm.network :forwarded_port, guest: 4567, host: 4567
  #antes de la actualizacion a Vagrant 1.5.3 funcionaba esto
  #config.vm.network :forwarded_port, id: 'ssh', guest: 22, host: 2222
  config.vm.network :forwarded_port, guest: 22, host: 2224

#  config.vm.boot_mode = :gui


  config.vm.synced_folder "#{edx_platform_mount_dir}", "/edx/app/edxapp/edx-platform", :create => true, nfs: true
  config.vm.synced_folder "#{forum_mount_dir}", "/edx/app/forum/cs_comments_service", :create => true, nfs: true
  config.vm.synced_folder "#{ora_mount_dir}", "/edx/app/ora/ora", :create => true, nfs: true

  config.vm.provider :virtualbox do |vb|
    vb.customize ["modifyvm", :id, "--memory", MEMORY.to_s]
    vb.customize ["modifyvm", :id, "--cpus", CPU_COUNT.to_s]

    # Allow DNS to work for Ubuntu 12.10 host
    # http://askubuntu.com/questions/238040/how-do-i-fix-name-service-for-vagrant-client
    vb.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
  end

  # Assume that the base box has the edx_ansible role installed
  # We can then tell the Vagrant instance to update itself.
  config.vm.provision "shell", inline: $script

end
```

FIGURA 3-16. CONFIGURACIÓN DE EDX DEVELOPER STACK EN VAGRANTFILE

3.3. edX Developer Stack

La edX Developer Stack, comúnmente llamada Devstack, es una instancia de *Vagrant* diseñada para el desarrollo de aplicaciones en local. Puede ejecutar tanto código como pruebas y la mayor parte del desarrollo puede hacerse en el entorno del *Host* [91]:

- Los repositorios Git son compartidos con el sistema *Host*, por lo que se puede utilizar el editor/IDE preferido.
- Se pueden cargar las páginas web en el sistema *Host* gracias a la instancia de *Vagrant* que está corriendo.

Como podemos comprobar en la imagen 3-17 del 'Vagrantfile' la Devstack tiene los siguientes componentes [92]:

- **edxapp**: que contiene la instancia de la plataforma edX con el LMS y el CMS del proyecto.
- **cs_comments_service**: sistema de comentarios independiente que soporta votaciones y comentarios así como foros de discusión.
- **ORA (*Open Response Assessor*)**: sistema para la evaluación de respuestas a preguntas abiertas (textos largos con imágenes adjuntas) que permite a los estudiantes puntuar tanto sus respuestas como las de sus compañeros.

```
Vagrant.configure("2") do |config|

  # Creates an edX devstack VM from an official release
  config.vm.box     = "gugelhupf-devstack"
  config.vm.box_url = "http://files.edx.org/vagrant-images/20140210-gugelhupf-devstack.box"

  config.vm.network :private_network, ip: "192.168.83.10"
  config.vm.network :forwarded_port, guest: 8000, host: 8000
  config.vm.network :forwarded_port, guest: 8001, host: 8001
  config.vm.network :forwarded_port, guest: 4567, host: 4567
  #antes de la actualizacion a Vagrant 1.5.3 funcionaba esto
  #config.vm.network :forwarded_port, id: 'ssh', guest: 22, host: 2222
  config.vm.network :forwarded_port, guest: 22, host: 2224

#  config.vm.boot_mode = :gui


  config.vm.synced_folder "#{edx_platform_mount_dir}", "/edx/app/edxapp/edx-platform", :create => true, nfs: true
  config.vm.synced_folder "#{forum_mount_dir}", "/edx/app/forum/cs_comments_service", :create => true, nfs: true
  config.vm.synced_folder "#{ora_mount_dir}", "/edx/app/ora/ora", :create => true, nfs: true

  config.vm.provider :virtualbox do |vb|
    vb.customize ["modifyvm", :id, "--memory", MEMORY.to_s]
    vb.customize ["modifyvm", :id, "--cpus", CPU_COUNT.to_s]

    # Allow DNS to work for Ubuntu 12.10 host
    # http://askubuntu.com/questions/238040/how-do-i-fix-name-service-for-vagrant-client
    vb.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
  end

  # Assume that the base box has the edx_ansible role installed
  # We can then tell the Vagrant instance to update itself.
  config.vm.provision "shell", inline: $script

end
```

FIGURA 3-17. COMPONENTES DE DEVSTACK

Cuando ejecutamos ciertos comandos en la consola es necesario indicar qué configuración queremos utilizar añadiendo algún parámetro. En nuestro caso, para que el sistema utilice la configuración Devstack, añadiremos el parámetro `--settings=devstack` para, por ejemplo, arrancar el servidor o realizar migraciones y lo indicaremos explícitamente para arrancar el lms o cms mediante los comandos `rake devstack[lms]` o `rake devstack[studio]` respectivamente.

3.4. Bases de Datos

edX utiliza tres tipos diferentes de bases de datos, *MySQL*, *SQLite* y *MongoDB*, y en particular la configuración Devstack utiliza solo dos de estos tres tipos, compartidos por el LMS y el CMS :

- ***MongoDB***: base de datos noSQL donde se almacena todo el contenido del curso y de los foros de discusión. El uso de esta base de datos es relativamente nuevo, anteriormente se utilizaban ficheros *XML*.
- ***MySql***: base de datos relacional donde se almacenan todos los datos de usuario.

3.4.1. MongoDB

FIGURA 3-18. LOGO DE *MONGODB*

MongoDB es un sistema de gestión de bases de datos NoSQL de código abierto (licencia GNUAGPL3.0) creado en 2007 orientado a documentos. Su nombre proviene de la palabra inglesa *humongous* [97], que significa enorme, y es de esquema libre, es decir, cada entrada o registro, a diferencia de los registros en las tablas de las bases de datos relacionales, puede tener atributos que pueden no estar presentes en otras entradas, y esos atributos pueden ser agregados, eliminados, modificados o renombrados en cualquier momento [98].

Los datos se guardan en documentos tipo *JSON* con un esquema dinámico llamado *BSON* (*Binary JSON*) y estos documentos en colecciones, podríamos decir que, si hacemos una comparación con las bases de datos relacionales, las tablas equivaldrían a las colecciones y los documentos serían los registros de la tabla.

La estructura de estos documentos es del tipo clave-valor (en inglés *key-value pairs*), separados por ':', donde la clave es el nombre del campo y el valor es su contenido [95]. También, *BSON*, guarda de forma explícita toda la información útil que permita búsquedas rápidas de datos, por ejemplo, las longitudes de los campos o los índices de los arrays., lo que hace que destaque por su velocidad y su sencillo sistema de consulta de contenidos [99].

En edX, como ya dijimos anteriormente, se utiliza este sistema para almacenar el contenido de los cursos creados en Studio así como el de los debates o foros de discusión. En particular, se utilizan dos bases de datos [31]:

- **xmodule**: contiene las definiciones y los metadatos. Aquí, la colección 'modulestore' guarda los documentos con el contenido y la información de los cursos.
- **xcontent**: contiene archivos que se hayan añadido al contenido del curso, como PDFs.

Al tener un esquema dinámico, la estructura de los documentos almacenados en la colección 'modulestore' varía en función de la información que contenga, es decir, si, por ejemplo, son los datos de un curso, no encontraremos los mismos atributos que si fuera una sección, una subsección o un problema. Pero sí que hay cuatro campos fijos:

[A partir de este punto nos referiremos a los distintos tipos de contenido como módulos]

- **`id`**: diccionario que almacena la localización del módulo. La *URL* se divide en partes y cada parte se almacena en un campo distinto.
- **`definition`**: diccionario que almacena los campos referentes al contenido del módulo.
- **`definition.children`**: lista de los localizadores (*URLs*) de los hijos del módulo.
- **`metadata`**: diccionario que almacena los campos referentes a la configuración del módulo.

En las imágenes 3-19, 3-20 y 3-21 podemos observar distintos campos en función del tipo de módulo:

```
    "_id" : {
            "tag" : "i4x",
            "org" : "uc3m",
            "course" : "CP01",
            "category" : "chapter",
            "name" : "7045ffa8f0044e24b48b105df291d01f",
            "revision" : null
    },
    "definition" : {
            "children" : [
                    "i4x://uc3m/CP01/sequential/d96835864b29489097877c111a8f34ad",
                    "i4x://uc3m/CP01/sequential/76b34f64463e4b4595f7f5cf7a1c4682",
                    "i4x://uc3m/CP01/sequential/cfda140a95d84d6e9178979c56e83df4"
            ],
            "data" : {

            }
    },
    "metadata" : {
            "start" : "2014-02-19T23:00:00Z",
            "display_name" : "Section 1"
    }
```


FIGURA 3-19. MÓDULO DE TIPO CHAPTER EN *MONGODB*


```
{
        "_id" : {
                "tag" : "i4x",
                "org" : "uc3m",
                "course" : "CP01",
                "category" : "problem",
                "name" : "aa75980c00434a919d421e23490447ee",
                "revision" : null
        },
        "definition" : {
                "data" : "<problem>\n<p>A checkboxes problem presents checkbox bu
>\n<choiceresponse>\n  <checkboxgroup direction=\"vertical\">\n    <choice correc
ice correct=\"true\">correct</choice>\n  </checkboxgroup>\n</choiceresponse>\n\n\
        },
        "metadata" : {
                "published_by" : NumberLong(1),
                "markdown" : "A checkboxes problem presents checkbox buttons for
 incorrect\n[x] correct\n",
                "display_name" : "3.3 - Checkboxes 1",
                "published_date" : [
                        2014,
                        6,
                        24,
                        20,
                        20,
                        12,
                        1,
                        175,
                        0
                ]
        }
}
```


FIGURA 3-20. MÓDULO DE TIPO PROBLEM EN *MONGODB*

```
{
        "_id" : {
                "tag" : "i4x",
                "org" : "uc3m",
                "course" : "CP01",
                "category" : "course",
                "name" : "2014_T1",
                "revision" : null
        },
        "definition" : {
                "children" : [
                        "i4x://uc3m/CP01/chapter/7045ffa8f0044e24b48b105df291d01f",
                        "i4x://uc3m/CP01/chapter/eab1a6edd43248a98a14bf1c5c42bf40",
                        "i4x://uc3m/CP01/chapter/a1748cb05b1b46bf9f984a1ec2787404"
                ],
                "data" : {
                        "wiki_slug" : "CP01"
                }
        },
        "metadata" : {
                "display_name" : "Prueba 1",
                "tabs" : [
                        {
                                "type" : "courseware",
                                "name" : "Courseware"
                        },
                        {
                                "type" : "course_info",
                                "name" : "Course Info"
                        },
                        {
                                "type" : "discussion",
                                "name" : "Discussion"
                        },
                        {
                                "type" : "wiki",
                                "name" : "Wiki"
                        },
                        {
                                "type" : "progress",
                                "name" : "Progress"
                        },
                        {
                                "type" : "static_tab",
                                "name" : "Recommend me!",
                                "url_slug" : "f4e1065fda654b6b910b62742afad761"
                        }
                ],
                "enrollment_start" : "2014-02-15T23:00:00Z",
                "discussion_topics" : {
                        "General" : {
                                "id" : "i4x-uc3m-CP01-course-2014_T1"
                        }
                },
                "start" : "2014-03-31T22:00:00Z",
                "enrollment_end" : "2014-09-29T22:00:00Z",
                "end" : "2014-12-30T23:00:00Z"
        }
}
```


FIGURA 3-21. MÓDULO DE TIPO COURSE EN *MONGODB*

En el fichero `/vagrant/edx-platform/docs/en_us/internal/persistence.rst` se describe en detalle cómo se representa cada tipo de contenido, su estructura y los posibles campos con su sintaxis. Y en el fichero `/vagrant/edx-platform/common/lib/xmodule/xmodule/modulestore/mongo.base.py` se definen las funciones que utilizaremos para recuperar los datos de la base de datos y poder utilizarlos en el código de nuestra aplicación.

3.4.2. MySQL

FIGURA 3-22. LOGO DE MYSQL

MySQL es un sistema de gestión de bases de datos relacional desarrollada por la empresa MySQL AB [100], fundada en 1995 por David Axmark, Allan Larsson y Michael Widenius. Esta empresa posee los derechos de la mayor parte del código, lo que hace que *MySQL* tenga licencia dual: una libre bajo la GNU GPL v2 y otra comercial para aquellas empresas que quieran darle uso en productos privativos.

Actualmente se utiliza en numerosos sitios web importantes, como Wikipedia[11], Google[12], Facebook[13], Twitter[14], Flickr[15] o YouTube[16].

[11] http://meta.wikimedia.org/wiki/Wikimedia_servers#System_architecture
[12] http://zurlocker.typepad.com/theopenforce/2005/12/googles_use_of_.html
[13] http://www.youtube.com/watch?v=Zofzid6xIZ4
[14] http://www.youtube.com/watch?v=5cKTP36HVgI
[15] http://www.mysql.com/customers/view/?id=720
[16] http://www.mysql.com/customers/view/?id=750

MySQL está escrito en C y C++ y el lenguaje que utiliza para realizar las consultas es *SQL* (*Structured Query Language*) [101]. *SQL* es el lenguaje más común para construir las consultas en los sistemas de gestión bases de datos relacionales, al ser un lenguaje de "alto nivel" permite una alta productividad en la codificación y la orientación a objetos y aprovecha la flexibilidad y potencia de este tipo de sistemas.

Algunas de las características que hacen de *MySQL* un sistema veloz, robusto y uno de los gestores con mejor rendimiento son las siguientes:

- Bajo coste en requerimientos para la creación de bases de datos ya que debido a su bajo consumo puede ser ejecutado en máquinas con pocos recursos.
- Facilidad de instalación y configuración.
- Alto nivel de portabilidad entre sistemas.
- Soporta una gran variedad de sistemas operativos.
- Garantiza seguridad en la conexión y en los datos mediante un sistema flexible de contraseñas y gestión de usuarios.
- Aprovecha la potencia de los sistemas multiproceso gracias a su implementación multihilo.
- Ofrece selección de mecanismos de almacenamiento con distinta velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones, etc.
- Amplio subconjunto del lenguaje *SQL*.
- Cada base de datos posee tres archivos: uno de estructura, uno de datos y uno de índices, con hasta 32 índices por tabla.
- Replicación, transacciones y claves foráneas.
- Búsqueda e indexación de campos de texto.
- Baja probabilidad de corromper datos.

En edX, como ya dijimos, este sistema de bases de datos se utiliza para almacenar toda la información referente a los usuarios: registros, inscripciones a los cursos, progreso, etc. En este proyecto utilizamos la base de datos llamada 'edxapp', donde se creará la tabla de nuestra aplicación (`recommender_student`) según el modelo de *Django* desarrollado. Las tablas de esta base de datos podemos verlas en la captura 3-23:

```
mysql> show tables;
+-----------------------------------------------+
| Tables_in_edxapp                              |
+-----------------------------------------------+
| auth_group                                    |
| auth_group_permissions                        |
| auth_permission                               |
| auth_registration                             |
| auth_user                                     |
| auth_user_groups                              |
| auth_user_user_permissions                    |
| auth_userprofile                              |
| bulk_email_courseauthorization                |
| bulk_email_courseemail                        |
| bulk_email_courseemailtemplate                |
| bulk_email_optout                             |
| celery_taskmeta                               |
| celery_tasksetmeta                            |
| certificates_certificatewhitelist             |
| certificates_generatedcertificate             |
| circuit_servercircuit                         |
| course_groups_courseusergroup                 |
| course_groups_courseusergroup_users           |
| course_modes_coursemode                       |
| courseware_offlinecomputedgrade               |
| courseware_offlinecomputedgradelog            |
| courseware_studentmodule                      |
| courseware_studentmodulehistory               |
| courseware_xmodulestudentinfofield            |
| courseware_xmodulestudentprefsfield           |
| courseware_xmoduleuserstatesummaryfield       |
| dark_lang_darklangconfig                      |
| django_admin_log                              |
| django_comment_client_permission              |
| django_comment_client_permission_roles        |
| django_comment_client_role                    |
| django_comment_client_role_users              |
| django_content_type                           |
| django_openid_auth_association                |
| django_openid_auth_nonce                      |
| django_openid_auth_useropenid                 |
| django_session                                |
| django_site                                   |
| djcelery_crontabschedule                      |
| djcelery_intervalschedule                     |
| djcelery_periodictask                         |
| djcelery_periodictasks                        |
| djcelery_taskstate                            |
| djcelery_workerstate                          |
| external_auth_externalauthmap                 |
| foldit_puzzlecomplete                         |
| foldit_score                                  |
| instructor_task_instructortask                |
| licenses_coursesoftware                       |
| licenses_userlicense                          |
```

```
| linkedin_linkedin                                  |
| notes_note                                         |
| notifications_articlesubscription                  |
| notify_notification                                |
| notify_notificationtype                            |
| notify_settings                                    |
| notify_subscription                                |
| psychometrics_psychometricdata                     |
| recommender_student                                |
| reverification_midcoursereverificationwindow       |
| shoppingcart_certificateitem                       |
| shoppingcart_order                                 |
| shoppingcart_orderitem                             |
| shoppingcart_paidcourseregistration                |
| shoppingcart_paidcourseregistrationannotation      |
| south_migrationhistory                             |
| splash_splashconfig                                |
| student_anonymoususerid                            |
| student_courseenrollment                           |
| student_courseenrollmentallowed                    |
| student_loginfailures                              |
| student_pendingemailchange                         |
| student_pendingnamechange                          |
| student_userstanding                               |
| student_usertestgroup                              |
| student_usertestgroup_users                        |
| track_trackinglog                                  |
| user_api_userpreference                            |
| verify_student_softwaresecurephotoverification     |
| waffle_flag                                        |
| waffle_flag_groups                                 |
| waffle_flag_users                                  |
| waffle_sample                                      |
| waffle_switch                                      |
| wiki_article                                       |
| wiki_articleforobject                              |
| wiki_articleplugin                                 |
| wiki_articlerevision                               |
| wiki_articlesubscription                           |
| wiki_attachment                                    |
| wiki_attachmentrevision                            |
| wiki_image                                         |
| wiki_imagerevision                                 |
| wiki_reusableplugin                                |
| wiki_reusableplugin_articles                       |
| wiki_revisionplugin                                |
| wiki_revisionpluginrevision                        |
| wiki_simpleplugin                                  |
| wiki_urlpath                                       |
+----------------------------------------------------+
100 rows in set (0.00 sec)
```

FIGURA 3-23. TABLAS DE LA BASE DE DATOS EDXAPP

4. Algoritmo de recomendación

En este capítulo se procederá a explicar de forma detallada el algoritmo de recomendación implementado para esta aplicación y se hará una especificación de los requisitos para la consecución de nuestro objetivo, obtener recomendaciones fiables para los alumnos de los cursos de la plataforma edX.

El algoritmo implementado en este proyecto está basado en los sistemas de filtrado colaborativo, explicados en el apartado 2.4.2., ya que hace predicciones sobre los problemas más apropiados para un alumno en un momento determinado del curso basándose en la experiencia de los compañeros con patrones similares de desempeño.

Los compañeros de curso son los colaboradores, sin embargo, en lugar de compartir los mismos patrones de evaluación con el usuario al que se va a hacer la recomendación, en este caso, la similitud entre el alumno y sus compañeros es calculada en función del número de problemas coincidentes realizados de forma satisfactoria. Para explicar el algoritmo de forma detallada partimos de los siguientes supuestos:

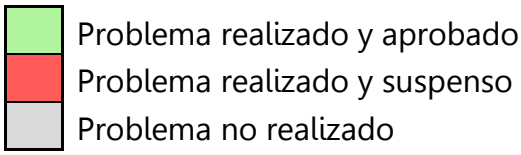
- Suponemos que tenemos $m+1$ alumnos registrados en un curso y n problemas en el mismo, $\{p_1, p_2, \dots, p_n\}$.
- El alumno a_0 es el alumno logueado en la plataforma y requiere una recomendación en un momento determinado del curso.
- El resto de alumnos, $\{a_1, a, \dots, a_m\}$, son los compañeros de curso de a_0 que actuarán de colaboradores.

Ilustraremos mediante un ejemplo el mecanismo del algoritmo. En la tabla 4-1 se recogen las similitudes y diferencias del alumno a_0 con sus compañeros de curso en el momento en que hace uso del recomendador. En nuestro ejemplo:

- m= 15 -> El alumno a_0 tiene 15 compañeros en el curso.
- n= 15 -> Hay 15 módulos de tipo `problem` en el curso.

TABLA 4-1. PROBLEMAS COINCIDENTES EN ESE MOMENTO DEL CURSO

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
p_1																
p_2																
p_3																
p_4																
p_5																
p_6																
p_7																
p_8																
p_9																
p_{10}																
p_{11}																
p_{12}																
p_{13}																
p_{14}																
p_{15}																
Nºproblemas aprobados coincidentes		**5**	4	**5**	3	2	**5**	1	4	**5**	**5**	**5**	3	**5**	2	4



Se comparan los problemas realizados por cada compañero del curso con los problemas realizados por el alumno a_0 y se obtiene el número de problemas aprobados en los que coinciden. En este caso observamos que el mayor número de problemas aprobados coincidentes es 5 y que son 7 los compañeros que coinciden en 5 problemas aprobados: $\{a_1, a_3, a_6, a_9, a_{10}, a_{11}, a_{13}\}$. A partir de ahora nos referiremos a ellos como 'compañeros más coincidentes'.

En la tabla 4-2 se representan los problemas en los que difiere cada uno de estos compañeros más coincidentes con el alumno a_0. Solo se tienen en cuenta los problemas que a_0 no haya realizado, los que haya realizado y suspendido no cuentan como problemas diferentes. Como el compañero más coincidente que más problemas ha realizado solo ha llegado hasta el p_{12} nos limitaremos a representar hasta ese problema.

TABLA 4-2. COMPAÑEROS MÁS COINCIDENTES Y MENOS DIFERENTES

	a_0	a_1	a_3	a_6	a_9	a_{10}	a_{11}	a_{13}
p_1								
p_2								
p_3								
p_4								
p_5								
p_6								
p_7								
p_8								
p_9								
p_{10}								
p_{11}								
p_{12}								
Nºproblemas diferentes		**2**	0	0	4	**2**	**2**	**2**

Problema realizado y aprobado
Problema realizado y suspenso
Problema no realizado
Problema que no se tiene en cuenta porque a_0 ya lo ha realizado aunque suspendido

En este punto descartamos los compañeros más coincidentes que no difieran en ningún problema ya que lo que buscamos son compañeros que tengan problemas susceptibles de ser recomendados. Quedan fuera del estudio $\{a_3, a_6\}$ ya que difieren en 0 problemas y continuaremos con los compañeros más coincidentes que difieren en menor número de problemas, $\{a_1, a_{10}, a_{11}, a_{13}\}$, a partir de ahora nos referiremos a ellos como 'compañeros más coincidentes y menos diferentes'. El alumno a_9 también queda excluido por el momento.

TABLA 4-3. NÚMERO DE VECES QUE SE REPITE CADA PROBLEMA SUSCEPTIBLE DE SER RECOMENDADO

	$\boldsymbol{a_0}$	a_1	a_{10}	a_{11}	a_{13}	Repeticiones
p_1						
p_2						
p_3						
p_4						
p_5						
p_6						
p_7						
p_8						
p_9						4
p_{10}						3
p_{11}						1

Problema realizado y aprobado
Problema realizado y suspenso
Problema no realizado
Problema que no se tiene en cuenta porque a_0 ya lo ha realizado aunque suspendido
Problema susceptible de ser recomendado

En la tabla 4-3 se calcula el número de veces que se ha realizado con éxito cada problema susceptible de ser recomendado por parte de los compañeros más coincidentes y menos diferentes. Observamos que los problemas ordenados según su popularidad quedan de la siguiente manera $\{p_9, p_{10}, p_{11}\}$ siendo p_9 el problema más repetido.

Por tanto, los problemas recomendados por parte de los compañeros más coincidentes y menos diferentes son: $\{\boldsymbol{p_9}, \boldsymbol{p_{10}}, \boldsymbol{p_{11}}\}$.

Sin embargo, se considera que para poder seguir avanzando en el curso es necesario que el alumno a_0 haya aprobado todos los problemas anteriores, por lo que, antes de recomendar problemas por parte de sus compañeros más coincidentes y menos diferentes recomendamos sus propios problemas suspensos, en este caso: $\{\boldsymbol{p_4}\}$.

En este momento tenemos cuatro recomendaciones, ordenadas: $\{\boldsymbol{p_4}, \boldsymbol{p_9}, \boldsymbol{p_{10}}, \boldsymbol{p_{11}}\}$.

La aplicación está configurada por defecto para almacenar 4 recomendaciones para cada alumno en la tabla 'recommender_student' de la base de datos *MySQL*, sin

embargo, este es un parámetro modificable y solo sería necesario cambiar el valor del parámetro `number` de la función `set_recommendations(user_id, course_id, number)` y realizar algunos cambios más en la función `get_recommendations(user_id, course_id, number)` (comentados en el código de la aplicación).

En caso de que se decidiera incrementar el número de recomendaciones a almacenar, por ejemplo a 6, y ya que no tenemos más que 4 problemas recomendados debemos recuperar a_9, el compañero más coincidente y más diferente que los menos diferentes que habíamos excluido anteriormente por diferir en 4 problemas en lugar de 2. Observamos que difiere en los problemas $\{p_9, p_{10}, p_{11}, p_{12}\}$, y que $\{p_9, p_{10}, p_{11}\}$ ya están en la lista de recomendaciones, luego $\{p_{12}\}$ pasa a ser un nuevo problema recomendado, quedando como recomendaciones y en este orden: $\{\boldsymbol{p_4}, \boldsymbol{p_9}, \boldsymbol{p_{10}}, \boldsymbol{p_{11}}, \boldsymbol{p_{12}}\}$.

Llegados a este punto y ya que no disponemos de más problemas susceptibles de ser recomendados, ni por ser problemas suspensos del alumno ni por colaboración de sus compañeros, la recomendación que nos falta se establece a '*None*': $\{\boldsymbol{p_4}, \boldsymbol{p_9}, \boldsymbol{p_{10}}, \boldsymbol{p_{11}}, \boldsymbol{p_{12}}, \boldsymbol{None}\}$.

El número de recomendaciones que queremos mostrar al alumno en la pestaña *Recommend Me!* también es un parámetro modificable. En la llamada a la función `get_recommendations(user_id, course_id, number)`, en el fichero *HTML*, se establece mediante el valor del parámetro `number`.

5. Análisis, diseño e implementación

En este capítulo se analizan los requisitos que se deben cumplir en base a las especificaciones de la plataforma edX, se hace un diseño de la arquitectura para la consecución de nuestro objetivo y se explican en detalle los pasos seguidos para la implementación de la aplicación desarrollada en este proyecto.

En el apartado 5.1 se estudia la base de datos en *MongoDB* y sus objetos y campos que nos van a ser útiles para el desarrollo de nuestra aplicación, así como las tablas de la base de datos en *MySQL* con las que vamos a interactuar.

En el apartado 5.2 se explican los pasos seguidos antes de comenzar a implementar nuestra aplicación, creación de un curso, registro de alumnos, etc. Y se comprueba el almacenamiento de la información en sus respectivas tablas. También se especifican los campos que va a tener la aplicación y se explica cómo se realiza la conexión con las bases de datos

Posteriormente se presenta el algoritmo de recomendación, se hace una breve descripción de cada una de las funciones implementadas y de su funcionalidad y se explica la lógica de la aplicación mediante diagramas de flujo.

5.1. Análisis

Una vez instalada la instancia de la plataforma edX y las herramientas necesarias, comenzamos haciendo un estudio de las bases de datos, tanto de la base de datos 'edxapp' en *MongoDB* como de las tablas de la base de datos 'edxapp' en *MySQL*. A continuación se explican y detallan los atributos y campos que vamos a utilizar en nuestra aplicación [102].

5.1.1. Atributo `id` de MongoDB

De la colección 'modulestore' de la base de datos 'edxapp' de *MongoDB* necesitamos recuperar exclusivamente los identificadores (`id`) de todos los módulos de la categoría `problem` pertenecientes a un determinado curso. Estos identificadores los guardaremos en una lista y serán utilizados en el algoritmo de recomendación de la aplicación.

Como indicamos en el apartado 3.4.1, este atributo es una *URL* formada por varias partes que se almacenan en distintos campos separados por '/':

- **`tag:`** convención basada en los planes para un dominio i4x.org.
- **`org:`** organización que ha creado en curso al que pertenece el módulo.
- **`course:`** número del curso al que pertenece el módulo.
- **`category:`** tipo de módulo.
- **`name:`** nombre del módulo.
- **`revision:`** si es una versión borrador (*draft*) o publicada.

Un ejemplo de identificador de módulo sería el siguiente:

```
i4x://uc3m/CP01/course/2014_T1
```

5.1.2. Tabla `auth_user`

Es una tabla integrada en el *framework* de *Django* para edX que contiene toda la información de acceso de los usuarios registrados y sus permisos.

```
mysql> describe auth_user;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | int(11)      | NO   | PRI | NULL    | auto_increment |
| username     | varchar(30)  | NO   | UNI | NULL    |                |
| first_name   | varchar(30)  | NO   |     | NULL    |                |
| last_name    | varchar(30)  | NO   |     | NULL    |                |
| email        | varchar(75)  | NO   | UNI | NULL    |                |
| password     | varchar(128) | NO   |     | NULL    |                |
| is_staff     | tinyint(1)   | NO   |     | NULL    |                |
| is_active    | tinyint(1)   | NO   |     | NULL    |                |
| is_superuser | tinyint(1)   | NO   |     | NULL    |                |
| last_login   | datetime     | NO   |     | NULL    |                |
| date_joined  | datetime     | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

FIGURA 5-1. CAMPOS DE LA TABLA `AUTH_USER`

En la captura 5-1 podemos ver todos los campos de la tabla `auth_user` de la base de datos 'edxapp' en *MySQL*:

- **`id`**: es la clave primaria (*Primary Key*) utilizada en las *URLs* que hacen referencia al usuario. Un usuario tiene el mismo `id` aquí y en la base de datos en *MongoDB*.
- **`username`**: es único para cada usuario registrado, no se puede cambiar y es la única información que pueden ver el resto de usuarios.
- **`first_name`**: no se utiliza. En su lugar se utiliza el nombre completo de usuario guardado en `auth_userprofile.name`.
- **`last_name`**: no se utiliza. En su lugar se utiliza el nombre completo de usuario guardado en `auth_userprofile.name`.

- **email**: la dirección de correo del usuario utilizada para acceder a la plataforma. Es única para cada usuario registrado y no es visible para el resto.
- **password**: una versión hash de la contraseña del usuario. *Django* 1.3 utiliza SHA1 y *Django* 1.4 y posteriores PBKDF2 con SHA256.
- **is_staff**: si su valor es 1 indica que es un miembro de la plantilla de edX con sus correspondientes privilegios o desarrolladores de edX que necesitan acceso para tareas de pruebas y desarrollo.
- **is_active**: su valor es 1 si el usuario ha activado su cuenta de usuario pinchando en el *link* que se le envió por correo. Una vez que se ha puesto a 1 solo puede ponerse a 0 de forma manual en caso de que se le prohíba el acceso al usuario por algún motivo.
- **is_superuser**: su valor es 1 para los administradores del sitio de *Django*.
- **last_login**: fecha y hora de la última vez que el usuario inició sesión.
- **date_joined**: fecha en la que se creó la cuenta (no en la que se activó).

5.1.3. Tabla `courseware_studentmodule`

Es una tabla que almacena el estado y la puntuación de cada usuario para cada módulo del curso. Se define como módulo cualquier pieza de contenido presente en el curso: un video, un *HTML*, un problema, etc. O, también, colecciones de módulos: secuencias, verticales, capítulos, etc. De hecho, el propio curso es un módulo como tal, y el resto de los módulos que contiene son sus hijos (*children*).

Cada fila de la tabla guarda el estado o la puntuación de un módulo para cada usuario, por lo tanto, habrá una fila por cada módulo y usuario, convirtiendo esta tabla en la tabla más grande de la base de datos. En el momento en el que el usuario carga la página, el sistema busca los módulos que deben ser renderizados y después busca el estado de esos módulos para ese usuario. Sin no hay ninguna entrada para ese usuario de ese problema se crea una nueva fila.

En la imagen 5-2 podemos ver los campos de la tabla `courseware_studentmodule` de la base de datos edxapp:

```
mysql> describe courseware_studentmodule;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| module_type | varchar(32)  | NO   | MUL | NULL    |                |
| module_id   | varchar(255) | NO   | MUL | NULL    |                |
| student_id  | int(11)      | NO   | MUL | NULL    |                |
| state       | longtext     | YES  |     | NULL    |                |
| grade       | double       | YES  | MUL | NULL    |                |
| created     | datetime     | NO   | MUL | NULL    |                |
| modified    | datetime     | NO   | MUL | NULL    |                |
| max_grade   | double       | YES  |     | NULL    |                |
| done        | varchar(8)   | NO   | MUL | NULL    |                |
| course_id   | varchar(255) | NO   | MUL | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

FIGURA 5-2. CAMPOS DE LA TABLA COURSEWARE_STUDENTMODULE

- **`id`**: es la clave primaria (*Primary Key*). Raramente se utiliza ya que la mayoría de las búsquedas en esta tabla se realizan mediante la tripleta (`course_id, student_id, module_id`).
- **`module_type`**: determina el tipo de módulo. La tabla 5-1 muestra los distintos tipos de módulos posibles.

TABLA 5-1. TIPOS DE MÓDULOS EXISTENTES (MODULE_TYPE)

Tipo	Descripción
chapter	Capítulo del curso.
combinedopenended	Módulo para evaluar cuestiones abiertas mediante autoevaluación, evaluación entre compañeros o aprendizaje máquina.
conditional	Impide el acceso a determinadas partes del curso si no se han completado otras primero.
course	Módulo de mayor nivel del cual desciende todo el contenido del curso.
lti	Del inglés *Learning Tools Interoperability.* Módulo para añadir una aplicación de aprendizaje externa.
peergrading	Problema evaluado por otros compañeros.
problem	Problema para el que el usuario puede enviar respuestas. En edX hay varios tipos.
problemset	Colección de problemas y materiales renderizado como una barra horizontal. Su uso es inconsistente y en ocasiones se utiliza un módulo sequential en su lugar.
randomize	Módulo que tiene varias opciones y estas son mostradas a cada alumno de forma aleatoria.
sequential	Colección de videos, problemas y otros materiales renderizados com una barra horizontal.
video	Módulo que muestra un archivo de video.
word_cloud	Módulo que genera un gráfico a partir de las palabras que introducen los usuarios.

El recomendador desarrollado en este proyecto se centra exclusivamente en los módulos de tipo `problem`. Los módulos de tipo `problem` que podemos encontrar en la plataforma edX son los siguientes [103]:

- ***Checkbox Problem***: los alumnos eligen una o más opciones de la lista de posibles respuestas.
- ***Dropdown Problem***: los alumnos eligen una respuesta de una lista desplegable de opciones.
- ***Multiple Choice Problem***: los alumnos seleccionan una respuesta de una lista de opciones.
- ***Numerical Input***: requieren una respuesta numérica que incluya números enteros, fracciones y algunas constantes y operadores matemáticos.
- ***Text Input Problem***: los alumnos deben introducir una respuesta corta en formato de texto.
- ***Circuit Schematic Builder Problem***: los alumnos crean y modifican circuitos en una cuadrícula interactiva.
- ***Custom JavaScript Problem***: problemas creados en *HTML* integrados en Studio mediante un *IFrame*.
- ***Drag and Drop Problem***: requiere que los alumnos arrastren texto u objetos a un punto específico de una imagen.
- ***Image Mapped Input Problem***: requiere que los alumnos hagan *click* en un punto específico de la imagen.
- ***Math Expression Input Problems***: requiere que los alumnos introduzcan una expresión matemática en modo texto.
- ***Problem with Adaptive Hint***: problemas que dan *feedback* o pistas a los alumnos. Pueden ser de tipo *Text Input* o *Multiple Choice Problems*.
- ***Problem Written in LaTeX***: problemas escritos en *LaTeX* convertidos al formato de edX. Es un prototipo

- ***Write-Your-Own-Grader Problem***: problemas que evalúan la respuesta de los alumnos utilizando un script de Python diseñado por el propio creador del curso.

- **`module_id`**: identificador único de cada módulo del curso. Es una *URL* de la forma `i4x://{org}/{course_num}/{module_type}/{module_name}`. En la tabla 5-2 se indican las partes de las que consta y un ejemplo de cada una:

TABLA 5-2. PARTES DE MODULE_ID

Parte	Ejemplo	Definición
`i4x://`		Convención basada en los planes para un dominio i4x.org.
`org`	MITx	Organización que ha creado el módulo.
`course_num`	3.091x	Número del curso al que pertenece este módulo.
`module_type`	problemset	Tipo de modulo.
`module_name`	Sample_Problems	Nombre que adjudicado por el creador del módulo. Si no se le ha dado un nombre, el Sistema se lo da basándose en el tipo y hash del contenido, por ejemplo problema_21b67856783209.

Podemos observar una correspondencia entre las partes que forman el campo `module_id` de esta tabla y el atributo `id` de la base de datos 'edxapp' de *MongoDB*:

- **`student_id`**: es una referencia a `auth_user.id`.

- **state**: campo de texto JSON donde distintos tipos de módulos (`course`, `chapter`, `problemset`, `sequential` y `combinedopenended`) pueden almacenar su estado de la manera que quieran.

- **grade**: valor de coma flotante que indica la nota sin ponderar que ha obtenido el estudiante en ese problema. Este campo solo lo utilizan los módulos de tipo `problem` y `selfassessment`.

- **created**: fecha en la que se creó la fila. Normalmente es la misma fecha en la que el usuario accedió a esa parte del contenido.

- **modified**: fecha en la que la fila fue actualizada por última vez. Normalmente se actualiza cuando hay un cambio de estado en el módulo.

- **max_grade**: valor de coma flotante que indica la máxima puntuación posible sin ponderar para ese problema. Este campo solo se utiliza para los módulos susceptibles de ser evaluados, el resto lo tienen a *NULL*.

- **done**: no se utiliza. Siempre tiene el valor `na`.

- **course_id**: identificador del curso al que pertenece el módulo de esa fila. Se representa con el formato {org}/{course}/{run}, por ejemplo UC3M/6.001x/Verano_2014. El mismo contenido (`module_id`) se puede utilizar en cursos distintos y hay que almacenarlo de forma separada para cada curso.

5.2. Diseño e implementación

5.2.1. Interacción con el CMS y el LMS

Para diseñar nuestra aplicación debemos empezar trabajando en la parte de CMS (Studio). Para ello iniciamos sesión con nuestro usuario, que tiene permisos de administrador, creamos un curso ficticio en Studio y añadimos algunos de los tipos de problemas detallados en el apartado 5.1.3. Después accedemos a *MongoDB* y mediante el comando:

```
db.modulestore.find({"_id.org":"uc3m",   "_id.category":"course",
"_id.course":"CP01"}).pretty()
```

buscamos un módulo de la organización 'uc3m', de categoría 'course' y de nombre 'CP01', y comprobamos que se ha creado correctamente. Obtenemos la salida mostrada en la captura de la imagen 5-3:

```
> db.modulestore.find({"_id.org": "uc3m", "_id.category": "course", "_id.course": "CP01"}).pretty()
{
        "_id" : {
                "tag" : "i4x",
                "org" : "uc3m",
                "course" : "CP01",
                "category" : "course",
                "name" : "2014_T1",
                "revision" : null
        },
        "definition" : {
                "children" : [
                        "i4x://uc3m/CP01/chapter/7045ffa8f0044e24b48b105df291d01f",
                        "i4x://uc3m/CP01/chapter/eab1a6edd43248a98a14bf1c5c42bf40",
                        "i4x://uc3m/CP01/chapter/a1748cb05b1b46bf9f984a1ec2787404"
                ],
                "data" : {
                        "wiki_slug" : "CP01"
                }
        },
        "metadata" : {
                "display_name" : "Prueba 1",
                "tabs" : [
                        {
                                "type" : "courseware",
                                "name" : "Courseware"
                        },
                        {
                                "type" : "course_info",
                                "name" : "Course Info"
                        },
                        {
                                "type" : "discussion",
                                "name" : "Discussion"
                        },
                        {
                                "type" : "wiki",
                                "name" : "Wiki"
                        },
                        {
                                "type" : "progress",
                                "name" : "Progress"
                        },
                        {
                                "type" : "static_tab",
                                "name" : "Recommend me!",
                                "url_slug" : "f4e1065fda654b6b910b62742afad761"
                        }
                ],
                "enrollment_start" : "2014-02-15T23:00:00Z",
                "discussion_topics" : {
                        "General" : {
                                "id" : "i4x-uc3m-CP01-course-2014_T1"
                        }
                },
                "start" : "2014-03-31T22:00:00Z",
                "enrollment_end" : "2014-09-29T22:00:00Z",
                "end" : "2014-12-30T23:00:00Z"
        }
}
```

FIGURA 5-3. CURSO DE PRUEBA CREADO VISTO EN *MONGODB*

Una vez que tenemos el curso necesitamos alumnos registrados en el mismo. Para ello hemos creado 14 alumnos ficticios. Comprobamos que se han creado correctamente y en la figura 5-4 vemos todos sus campos accediendo a la tabla `auth_user`, explicada en el apartado 5.1.2:

```
mysql> select * from auth_user;
+----+----------+------------+-----------+----------------------------+-----------------------------------------------------
------------------------------+----------+-----------+--------------+--------------------+----------------------+
| id | username | first_name | last_name | email                      | password
                              | is_staff | is_active | is_superuser | last_login         | date_joined          |
+----+----------+------------+-----------+----------------------------+-----------------------------------------------------
------------------------------+----------+-----------+--------------+--------------------+----------------------+
|  1 | Alma     | Alma       | Collado   | alma.collado@gmail.com     | pbkdf2_sha256$10000$rgvAIjVKmkkR$Geo7q2J0kIqrSu
HobEOfjSIAYPs/v2/gEqPgdovc5W8= |        1 |         1 |            1 | 2014-08-24 12:58:09 | 2014-04-03 19:21:29 |
|  2 | Daniel   | Daniel     | Sanz      | daniel.sanz@gmail.com      | pbkdf2_sha256$10000$daEl9QBauzH6$u3qOKbdtio/1Pp
VayqP6silDWlty8NW0FNlDdGiCccU= |        0 |         1 |            0 | 2014-06-25 09:18:52 | 2014-04-03 20:32:45 |
|  3 | Marta    | Marta      | Fern□ndez | marta.fernandez@gmail.com  | pbkdf2_sha256$10000$m6P2nhDTJzO1$pG3LjJMV7cxeM3
XgPns5QdYbmbknLtdqRsOL2kHqLWI= |        0 |         1 |            0 | 2014-06-25 15:37:16 | 2014-04-03 20:33:27 |
|  4 | Noelia   | Noelia     | Segura    | noelia.segura@gmail.com    | pbkdf2_sha256$10000$dikh0duvPdP4$dJBHiuZTS1QjLY
ZaoMxTRgxilewqMO94dINLgI8u93s= |        0 |         1 |            0 | 2014-06-25 15:21:24 | 2014-04-03 22:01:52 |
|  5 | Miguel   | Miguel     | Mancha    | miguel.mancha@gmail.com    | pbkdf2_sha256$10000$05wEpiRuWKsx$LOvpcrTC9a2KZz
gHGNyTXJtgddqLml8YXiCaWOuYask= |        0 |         1 |            0 | 2014-06-25 15:29:09 | 2014-04-03 22:02:44 |
|  6 | Santiago | Santiago   | S□nchez   | santiago.sanchez@gmail.com | pbkdf2_sha256$10000$mKQ80v9t4uvT$lBG8AU3LRy+H7M
+tWlZaVU/GXh8IjYNRvaDqlOZtv+8= |        0 |         1 |            0 | 2014-06-25 10:49:49 | 2014-04-03 22:03:17 |
|  7 | Nacho    | Nacho      | Foche     | nacho.foche@gmail.com      | pbkdf2_sha256$10000$RBIcrAvOv7yb$7LlTCDFlaWKs/3
koe9tieEyV55XFiaZx7j6yFiVr5WI= |        0 |         1 |            0 | 2014-06-25 12:33:44 | 2014-04-03 22:03:49 |
|  8 | Raquel   | Raquel     | Gonz□lez  | raquel.gonzalez@gmail.com  | pbkdf2_sha256$10000$FIXZsilsKovg$Vf6wwNd1ZHzVNy
pnApoYkrapxfxaBOwht+yaUEC1PFo= |        0 |         1 |            0 | 2014-06-25 12:26:14 | 2014-04-03 22:04:25 |
|  9 | Laura    | Laura      | P□rez     | laura.perez@gmail.com      | pbkdf2_sha256$10000$j4YnYqkZNKNC$OxOq13VhgYkTOt
aW7Kvv/qfsplP1+dMXECP/m+MWb4I= |        0 |         1 |            0 | 2014-06-25 12:26:54 | 2014-04-03 22:04:56 |
| 10 | Carlos   | Carlos     | Garc□a    | carlos.garcia@gmail.com    | pbkdf2_sha256$10000$sepUumeEx6Sj$QrdXwwRI2wgUjy
Z+mHA8x7HdGLBEkB16SvSIMd/JIo4= |        0 |         1 |            0 | 2014-06-25 12:30:25 | 2014-04-03 22:05:37 |
| 11 | Alejandro | Alejandro  | Montes    | alejandro.montes@gmail.com | pbkdf2_sha256$10000$IGcRGmBn0fCn$z6DH1KNwiVcVLi
8aF0TmoJihe8UqEmVEDweBrgV+Hjo= |        0 |         1 |            0 | 2014-06-25 12:35:14 | 2014-04-03 22:06:59 |
| 12 | Borja    | Borja      | Mesas     | borja.mesas@gmail.com      | pbkdf2_sha256$10000$CyI9n6Tt8bOQ$jNjki8POuLZA3y
UMh705v22dlftnjJHtIETv128Vz9A= |        0 |         1 |            0 | 2014-06-25 12:39:35 | 2014-04-03 22:08:03 |
| 13 | Maria    | Mar□a      | Pereira   | maria.pereira@gmail.com    | pbkdf2_sha256$10000$4OLDFGel3g68$1HYU7QE6DkeT/B
a/ugRCWR0wbcb8RBR1Wu3E3bBA5k4= |        0 |         1 |            0 | 2014-04-03 22:22:10 | 2014-04-03 22:09:10 |
| 14 | Cristina | Cristina   | Rodr□guez | cristina.rodriguez@gmail.com | pbkdf2_sha256$10000$zcIX0HX8to8a$/fBPp6p3jBLH2G
ews81EWYuf47BYAqw6wTWDlQOTkTI= |        0 |         1 |            0 | 2014-06-25 15:22:00 | 2014-04-03 22:10:08 |
| 15 | Jose     | Jose       | Heras     | jose.heras@gmail.com       | pbkdf2_sha256$10000$x7cMb9YjE4Rq$wdexH8hMzmBU1B
uYDuJe8qIT65GEVDtrgx4V49PzjdI= |        0 |         1 |            0 | 2014-06-25 13:08:11 | 2014-04-03 22:10:48 |
+----+----------+------------+-----------+----------------------------+-----------------------------------------------------
------------------------------+----------+-----------+--------------+--------------------+----------------------+
15 rows in set (0.01 sec)
```

FIGURA 5-4. CONTENIDO DE LA TABLA AUTH_USER CON LOS ALUMNOS FICTICIOS CREADOS

Observamos que el primer usuario que aparece en la tabla es el administrador de Django, ya que tiene el campo `is_superuser` a 1. Este usuario es el encargado de activar las cuentas de los alumnos, es decir, acceder a la parte de administración de Django de *Auth>Users* y poner el campo `is_active` de cada uno de ellos a `1`.

Llegados a este punto ya podemos trabajar en la parte del LMS y acceder a la cuenta de cada alumno creado, con su usuario y contraseña, y empezar el curso resolviendo algunos problemas de forma aleatoria para poder ir implementando nuestra

aplicación y comprobar su funcionamiento. En la imagen 5-5 vemos el ejemplo de una entrada en la tabla `courseware_studentmodule`:

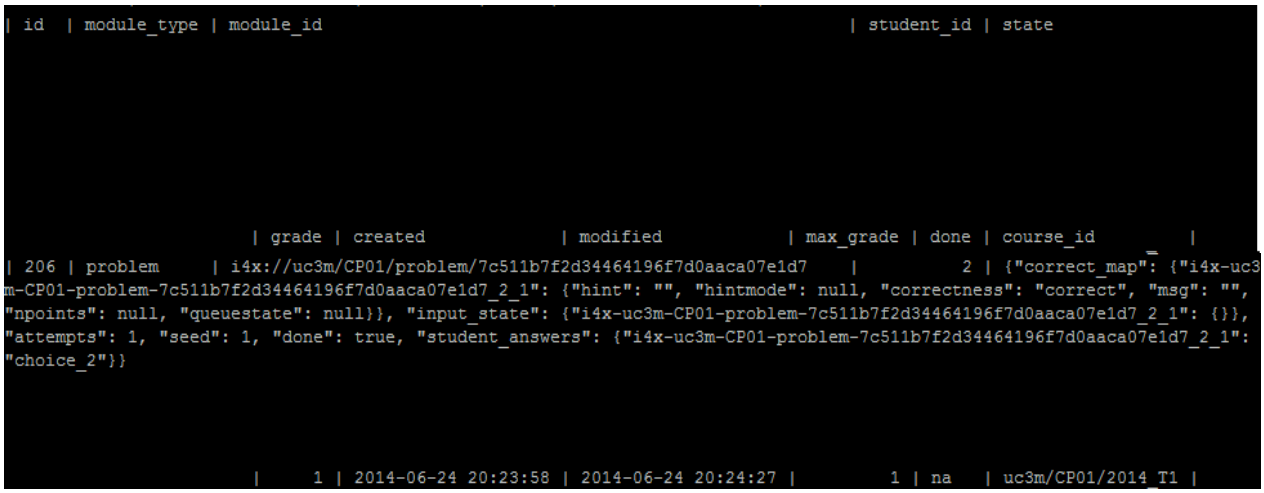


FIGURA 5-5. EJEMPLO DE UNA ENTRADA EN LA TABLA COURSEWARE_STUDENTMODULE

Si desglosamos cada uno de los campos que aparecen en la imagen anterior:

- **id=206**: identificador generado de forma automática. No se utiliza.
- **module_type=problem**: el módulo es de tipo problema.
- **module_id=i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7**: identificador del módulo.
- **student_id=2**: Identificador del usuario que ha realizado este problema. Si miramos En la tabla `auth_user` se corresponde con el alumno de `id=2`, cuyo `username` es Daniel.
- **state={"correct_map":{"i4x-uc3m-CP01-problem-7c511b7f2d34464196f7d0aaca07e1d7_2_1":{"hint":"","hintmode": null, "correctness": "correct", "msg": "", "npoints": null, "queuestate": null}}, "input_state": {"i4x-uc3m-CP01-problem-7c511b7f2d34464196f7d0aaca07e1d7_2_1":{}}, "attempts": 1, "seed": 1, "done": true, "student_answers": {"i4x-uc3m-CP01-problem-7c511b7f2d34464196f7d0aaca07e1d7_2_1":"choice_2"}**:

variables de estado del problema, como el número de intentos, utilización de pistas, respuestas del alumno Daniel, etc.

- **`grade=1`**: nota obtenida por el alumno Daniel en ese problema.
- **`created=2014-06-24 20:23:58`**: fecha en la que Daniel accedió al problema por primera vez.
- **`modified=2014-06-24 20:24:27`**: fecha de la última modificación por parte del alumno Daniel.
- **`max_grade=1`**: máxima nota posible en ese problema.
- **`done=na`**: No se utiliza.
- **`course_id=uc3m/CP01/2014_T1`**: Nombre del curso al que pertenece el problema.

5.2.2. Estructura y Parámetros de la aplicación

El propósito de este proyecto es la creación de un recomendador de problemas para los alumnos de un curso, por lo tanto, para desarrollar nuestro código necesitamos los siguientes parámetros:

- **Identificador del alumno**: campo `id` en la tabla `auth_user` que se corresponde con el campo `student_id` en la tabla `courseware_studentmodule`. Este parámetro se le asigna al alumno de forma automática cuando se registra en la plataforma y es único e invariable.
- **Identificador del curso**: campo `course_id` en la tabla `courseware_studentmodule`. Un alumno puede estar registrado en varios cursos a la vez, por lo que hay que tener en cuenta el nombre del curso a la hora de buscar los problemas a recomendar.
- **Tipo de módulo**: campo `module_type` de la tabla `courseware_studentmodule`. En la aplicación desarrollada solo se recomiendan módulos de tipo `problem`.

- **Identificador del módulo**: campo `module_id` de la tabla `courseware_studentmodule`. Para identificar los problemas.
- **Nota obtenida en el problema**: campo `grade` de la tabla `courseware_studentmodule`. Junto con el campo `max_grade` se utiliza para calcular si el alumno ha aprobado o no el problema.
- **Nota máxima a la que se puede optar en el problema**: campo `max_grade` de la tabla `courseware_studentmodule`. Junto con el campo `grade` se utiliza para calcular si el alumno ha aprobado o no el problema.

Una vez analizadas las tablas de 'edxapp' y estudiado sus campos y la relación entre ellos, hemos determinado los parámetros que debemos conocer a la hora de hacer una recomendación. De entre esos parámetros, hay tres principales necesarios para la creación de nuestro modelo de *Django*:

```
class Student(models.Model):

    user_id = models.ForeignKey(User, db_column='user_id', null=False, blank=False)
    course_id = models.CharField(max_length=250, db_column='course_id', null=True, blank=True)
    module_id = models.CharField(max_length=250, db_column='module_id', null=True, blank=True)
```

FIGURA 5-6. MODELO DE *DJANGO* RECOMMENDER_STUDENT

En la imagen 5-6 se detallan los tres campos de nuestro modelo, que equivalen a las columnas de la tabla `recommender_student`:

- **`user_id`**: identificador del alumno al que corresponde la recomendación de esta entrada de la tabla. Referencia al `id` de la tabla `auth_user`.
- **`course_id`**: identificador del curso del que se está haciendo la recomendación al alumno de esta entrada de la tabla.
- **`module_id`**: identificador del módulo tipo `problem` recomendado al alumno de esta entrada de la tabla.

En la imagen 5-7 podemos ver las cabeceras de las columnas de nuestra tabla y otra columna llamada `id`, generada automáticamente por el sistema para actuar de *Primary Key* pero que no se va a utilizar.

```
mysql> describe recommender_student;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_id   | int(11)      | NO   | MUL | NULL    |                |
| course_id | varchar(250) | YES  |     | NULL    |                |
| module_id | varchar(250) | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

FIGURA 5-7. CAMPOS DEL MODELO RECOMMENDER_STUDENT

La estructura de la aplicación, con los ficheros y directorios creados, se detalla a continuación:

- `/vagrant/edx-platform/lms/djangoapps >` dentro de la carpeta djangoapps se encuentran todas las aplicaciones de Django para la plataforma edX, por lo tanto aquí creamos nuestra nueva aplicación. Al ejecutar el comando: `python manage.py startapp recommender`, se crea una nueva carpeta que contendrá nuestra aplicación: recommender.

- `/vagrant/edx-platform/lms/djangoapps/recommender >` como vimos en el apartado 3.1.4 se han creado varios archivos de forma automática : `__init__.py`, `models.py`, `tests.py` y `views.py`. Ya hemos visto que en `models.py` se crean los modelos de nuestra aplicación, figura 5-6; `tests.py` y `views.py` no los vamos a modificar pero creamos otro fichero llamado `data.py`, donde implementa nuestro algoritmo de recomendación y establecemos las conexiones para recuperar la información requerida de las bases de datos, tanto de *MongoDB* como de *MySQL*. Todo esto se detalla en los siguientes apartados.

  En caso de que se quisieran gestionar las recomendaciones de forma manual desde el administrador de Django sería necesario crear un nueva archivo `recommender/admin.py` para indicarle al administrador que los objetos de tipo

`Student` tienen una interfaz de administración. Para nuestro proyecto no lo hemos creído necesario ya que las recomendaciones son el resultado de la ejecución del algoritmo de recomendación que detallaremos más adelante, y no algo que se deba gestionar de forma manual desde el administrador. Pero puede considerarse como una posible mejora futura si se desarrolla alguna nueva funcionalidad para la que sea necesaria.

- `/vagrant/edx-platform/lms/djangoapps/recommender/migrations` > una vez que se haya realizado la primera migración se crea esta carpeta de forma automática y almacena los ficheros Python donde se guarda la información de cada vez que se realiza algún cambio en nuestros modelos.

- `/vagrant/edx-platform/lms/templates/courseware` > en esta carpeta guardamos nuestro fichero *HTML* `(recommendations.html)` que define la apariencia de la pestaña al ser cargada en el navegador y desde donde se ejecuta el algoritmo de recomendación. Nuestra aplicación aparece en una `static_tab` dentro de la plataforma, por lo que para que sea mostrada en el navegador tenemos que realizar los siguientes pasos:

  - En Studio debemos crear una nueva `static_tab`: *Static Tab* > *Settings* > *Display Name* > *Reommend Me!*

  - Referenciar nuestro fichero *HTML* en la función `static_tab(request, course_id, tab_slug)` del fichero `/vagrant/edx-platform/lms/djangoapps/courseware/views.py`. Para ello lo modificamos de la siguiente manera:

```
return render_to_response('courseware/recommendations.html',
                {'course': course,
                 'tab': tab,
                 'tab_contents': contents,
                 'staff_access': staff_access, })
```

5.2.3. Conexión con MongoDB

Para explicar cómo se realiza la conexión con la base de datos de *MongoDB* utilizamos como ejemplo nuestra conexión con la base de datos 'edxapp' de *MongoDB* para obtener los documentos que hay en la colección 'modulestore':

```
1  # MongoDB connection
2  from pymongo import Connection
3
4  edxappdb = "edxapp"
5  connection = Connection()
6  db = connection[edxappdb]
7  mongo_modulestore = db['modulestore']
```

FIGURA 5-8. CONEXIÓN CON LA BASE DE DATOS DE *MONGODB*

Analizamos línea a línea la imagen 5-8:

- **Línea 2**: Importamos el módulo `Connection` de la librería `pymongo`.
- **Línea 4**: Almacenamos en una variable (`edxappdb`) el nombre de la base de datos a la que nos queremos conectar, en este caso 'edxapp'.
- **Línea 5**: Establecemos una conexión local. En caso de querer conectarnos a un servidor remoto los parámetros serían:

  `Connection('servidor', 'puerto')`.
- **Línea 6**: Obtenemos un objeto que referencia a la base de datos. También puede utilizarse:

  `db = connection.edxappdb`
- **Línea 7**: Obtenemos la colección con la que vamos a trabajar. También puede utilizarse:

  `mongo_modulestore = db.modulestore`

Una vez que tenemos la colección que vamos a utilizar para poder acceder a los documentos y recuperar la información requerida utilizamos las funciones propias de *MongoDB* para realizar consultas[17]. En la imagen 5-9 se muestra como ejemplo una función de nuestra aplicación, que se devuelve una lista con todos los problemas de un curso cuyo identificador se pasa por parámetro:

```
# Function that returns all the problems for the course with 'course_id'
def get_course_problems(course_id):

    parts = course_id.split('/')
    module_list = []
    for module in mongo_modulestore.find({'_id.course': parts[1], '_id.category':{'$in':['problem']}}):
        module_list.append(module)

    return module_list
```

FIGURA 5-9. FUNCIÓN QUE ILUSTRA LA OBTENCIÓN DE INFORMACIÓN DE *MONGODB*

En este caso utilizamos la función `find()` para recuperar los módulos de tipo `problem` del curso con el identificador `course_id` pasado por parámetro.

5.2.4. Conexión con MySQL

Para explicar cómo se realiza la conexión con la base de datos *MySQL* utilizamos como ejemplo nuestra conexión con la base de datos `edxapp` de *MySQL*:

[17] http://www.mongodb.org/

```
1   # MySQL database connection
2   import MySQLdb
3
4   cnx = MySQLdb.connect(user = 'root',db = 'edxapp')
5   cursor = cnx.cursor()
6   cursor.execute("SHOW TABLES LIKE 'courseware_studentmodule'")
7   result = cursor.fetchone()
8
9   cursor.close()
10  cnx.close()
```

FIGURA 5-10. CONEXIÓN CON LA BASE DE DATOS DE *MYSQL*

Analizamos línea a línea la imagen 5-10:

- **Línea 2**: Importamos el paquete `MySQLdb`.
- **Línea 4**: Mediante la función `connect(user,db)` del paquete `MySQLdb` creamos una conexión con usuario 'root' a la base de datos 'edxapp'.
- **Línea 5**: Creamos un `cursor` para realizar las operaciones en la base de datos.
- **Línea 6**: Ejecutamos la consulta con el método `execute()` que nos proporciona el `cursor`. En este caso es buscar la tabla de nombre `courseware_studentmodule`.
- **Línea 7**: Extraemos un solo resultado utilizando el método `fetchone()`.
- **Línea 9:** Cerramos el `cursor` una vez realizada la consulta.
- **Línea 10:** Nos desconectamos de la base de datos.

La imagen anterior es un ejemplo de cómo se conecta y desconecta de la base de datos y la consulta realizada lista las tablas de nombre `courseware_studentmodule`. El resultado de la sentencia `SHOW TABLES` se almacena en la variable `result`, que es la que hay que manipular según nuestras necesidades.

5.2.5. Implementación del algoritmo de recomendación

Una vez que hemos identificado los campos necesarios de las bases de datos y realizado las conexiones para la recuperación de los mismos procedemos a detallar el algoritmo de recomendación implementado para nuestra aplicación.

A continuación, en las tablas de la 5-3 a la 5-20, se presenta de forma esquemática cada una de las funciones desarrolladas con sus parámetros de entrada y salida y una breve descripción de su funcionalidad y en el apartado 5.2.6 se incluyen los diagramas de flujo que se corresponden con cada una de las mismas.

TABLA 5-3. FUNCIÓN GET_COURSE_PROBLEMS (COURSE_ID)

Nombre	`get_course_problems (course_id)`
Parámetros de entrada	`course_id`: identificador del curso
Descripción	Accede a la base de datos de *MongoDB* y obtiene todos los problemas contenidos en el curso
Salida	Lista de módulos tipo `problem`

* **Ver diagrama de flujo en la figura 5-11**

TABLA 5-4. FUNCIÓN GET_COURSE_PROBLEMS_ID (COURSE_ID)

Nombre	`get_course_problems_id (course_id)`
Parámetros de entrada	`course_id`: identificador del curso
Descripción	Forma los identificadores de los módulos tipo `problem` a partir de sus atributos
Salida	Lista de `module_ids`

* **Ver diagrama de flujo en la figura 5-12**

TABLA 5-5. FUNCIÓN GET_DISPLAY_NAME (MODULE_ID)

Nombre	`get_display_name (module_id)`
Parámetros de entrada	`module_id`: identificador del problema
Descripción	Accede a la base de datos *MongoDB* y obtiene el nombre del problema cuyo identificador hemos pasado por parámetro.
Salida	`String`

* **Ver diagrama de flujo en la figura 5-13**

TABLA 5-6. FUNCIÓN GET_GRADED_PROBLEMS (USER_ID, COURSE_ID)

Nombre	`get_graded_problems (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene de la tabla `courseware_studentmodule` de *MySQL* los identificadores de los problemas realizados por el alumno en el curso
Salida	Lista de module_ids

* **Ver diagrama de flujo en la figura 5-14**

TABLA 5-7. FUNCIÓN GET_IDS (USER_ID, COURSE_ID)

Nombre	`get_ids (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los compañeros de curso del alumno
Salida	Lista de `user_ids`

* **Ver diagrama de flujo en la figura 5-15**

TABLA 5-8. FUNCIÓN `GET_PASSED_PROBLEMS (USER_ID, COURSE_ID)`

Nombre	`get_passed_problems (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los problemas aprobados por el alumno en el curso calculando si la nota obtenida es mayor que la mitad de la nota posible
Salida	Lista de `module_ids`

* **Ver diagrama de flujo en la figura 5-16**

TABLA 5-9. FUNCIÓN `GET_FAILED_PROBLEMS (USER_ID, COURSE_ID)`

Nombre	`get_failed_problems (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los problemas suspensos del alumno a partir de los realizados y los aprobados
Salida	Lista de `module_ids`

* **Ver diagrama de flujo en la figura 5-17**

TABLA 5-10. FUNCIÓN `GET_CLASSMATES_PASSED_PROBLEMS(USER_ID, COURSE_ID)`

Nombre	`get_classmates_passed_problems(user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los problemas aprobados por cada uno de los compañeros de curso del alumno
Salida	Diccionario formado por {`user_id` de compañero: su lista de problemas aprobados}

* **Ver diagrama de flujo en la figura 5-18**

TABLA 5-11. FUNCIÓN `GET_NUMBER_OF_PASSED_COINCIDENCES(USER_ID, COURSE_ID)`

Nombre	`get_number_of_passed_coincidences(user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene el número de problemas aprobados en los que coincide cada compañero de curso con el alumno
Salida	Diccionario formado por {`user_id` de compañero: número de problemas aprobados coincidentes}

* **Ver diagrama de flujo en la figura 5-19**

TABLA 5-12. FUNCIÓN GET_PASSED_COINCIDENCES(USER_ID, COURSE_ID)

Nombre	`get_passed_coincidences(user_id,course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los problemas aprobados en los que coincide cada compañero de curso con el alumno
Salida	Diccionario formado por {`user_id` de compañero: lista de problemas aprobados coincidentes}

* **Ver diagrama de flujo en la figura 5-20**

TABLA 5-13. FUNCIÓN GET_MOST_COINCIDENT(USER_ID, COURSE_ID)

Nombre	`get_most_coincident (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los compañeros de curso que coinciden en más problemas aprobados con el alumno
Salida	Lista de `user_ids`

* **Ver diagrama de flujo en la figura 5-21**

TABLA 5-14. FUNCIÓN GET_NUMBER_OF_DIFFERENCES (USER_ID, COURSE_ID)

Nombre	`get_number_of_differences (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene el número de problemas en los que difiere el alumno con los compañeros de curso con mayor número de problemas aprobados coincidentes
Salida	Diccionario formado por {`user_id` de compañero más coincidente: número de problemas en los que difieren}

* **Ver diagrama de flujo en la figura 5-22**

TABLA 5-15. FUNCIÓN GET_LEAST_DIFFERENT (USER_ID, COURSE_ID)

Nombre	`get_least_different (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los compañeros de curso con mayor número de problemas aprobados coincidentes y menor número de diferentes
Salida	Lista de `user_ids`

* **Ver diagrama de flujo en la figura 5-23**

TABLA 5-16. FUNCIÓN GET_RECOMMENDED_PROBLEMS(USER_ID, COURSE_ID)

Nombre	`get_recommended_problems(user_id,course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Obtiene los identificadores de los compañeros de curso con mayor número de problemas aprobados coincidentes y menor número de diferentes y los problemas en los que difieren
Salida	Diccionario formado por {`user_id` de compañero más coincidente y menos diferente: problemas en los que difiere con el alumno}

* **Ver diagrama de flujo en la figura 5-24**

TABLA 5-17. FUNCIÓN EXTRACT_AND_COUNT(USER_ID, COURSE_ID)

Nombre	`extract_and_count (user_id, course_id)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
Descripción	Cuenta el número de compañeros que han aprobado cada problema en el que difieren con el alumno
Salida	Diccionario formado por {identificador del problema diferente: número de compañeros que han aprobado el problema}

* **Ver diagrama de flujo en la figura 5-25**

TABLA 5-18. FUNCIÓN GET_BEST_RECOMMENDATIONS(USER_ID, COURSE_ID, NUMBER)

Nombre	`get_best_recommendations(user_id,course_id,number)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
	`number`: número de problemas que queremos guardar en la tabla de recomendaciones
Descripción	Obtiene los problemas recomendados para el alumno en función de sus problemas suspensos, las repeticiones de los problemas en los que difiere con sus compañeros de curso y otros problemas aprobados por compañeros coincidentes
Salida	Lista de `module_ids`

* **Ver diagrama de flujo en la figura 5-26**

TABLA 5-19. FUNCIÓN SET_RECOMENDATIONS(USER_ID, COURSE_ID, NUMBER)

Nombre	`set_recommendations (user_id, course_id, number)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
	`number`: número de problemas que queremos guardar en la tabla de recomendaciones
Descripción	Almacena en la tabla `recommender_student` el número de recomendaciones requerido para el alumno en el curso
Salida	Sin salida

* **Ver diagrama de flujo en la figura 5-27**

Nota: *el parámetro `number`, que indica el número de recomendaciones para el alumno en el curso almacenadas en `recommender_student`, ha sido fijado en 4 pero se puede modificar en caso de que sea necesario.*

TABLA 5-20. FUNCIÓN GET_RECOMMENDATIONS (USER_ID, COURSE_ID, NUMBER)

Nombre	`get_recommendations (user_id, course_id, number)`
Parámetros de entrada	`user_id`: identificador del alumno
	`course_id`: identificador del curso
	`number`: número de problemas que queremos guardar en la tabla de recomendaciones
Descripción	Recupera de la tabla `recommender_student` el número de recomendaciones indicado por el parámetro `number`
Salida	Lista de module_ids

* **Ver diagrama de flujo en la figura 5-28**

> ***Nota**: el parámetro `number` indica el número de recomendaciones para el alumno en el curso que queremos recuperar de la tabla `recommender_student` para mostrarlas en la pestaña Recommend Me!. En la tabla hay 4 recomendaciones por alumno y curso, por lo que podremos elegir mostrar 1, 2, 3 o 4. Este parámetro se modifica en el fichero HTML de la aplicación, donde se llama a esta función.*

Adicionalmente se ha implementado una función de depuración, `debug_mode(user_id, course_id, number)`, que nos permite comprobar el correcto funcionamiento de la aplicación para los distintos casos de prueba. Esto se comprobará en el apartado 6 con varios ejemplos.

5.2.6. Lógica de la aplicación

El proceso que se realiza desde que el alumno selecciona la pestaña de nuestra aplicación, *Recommend Me!,* hasta que se muestran los problemas recomendados es el siguiente:

1. El alumno (`user_id`) registrado en un curso (`course_id`) está siguiendo el mismo y selecciona la pestaña *Recommend Me!*
2. Automáticamente se ejecuta el algoritmo de recomendación siguiendo estos pasos:

    1. Se accede a la base de datos *MySQL* y de la tabla 'courseware_studentmodule' se obtienen los identificadores de los problemas que ha realizado el alumno logueado (`user_id`) en el curso (`course_id`).
    2. Una vez que se tienen los problemas realizados por el alumno en el curso, se calcula cuáles están aprobados y cuáles están suspensos. Para ello se tiene en cuenta la nota obtenida y la nota máxima posible de cada problema.
    3. Como el algoritmo basa sus recomendaciones en las similitudes con los compañeros de curso es necesario obtener el identificador de usuario de cada uno de ellos.
    4. Una vez que tenemos los identificadores de los compañeros de curso necesitamos los identificadores de los problemas que han aprobado para poder calcular la similitud con el alumno logueado.
    5. Calculamos la similitud entre el alumno y cada uno de sus compañeros y nos quedamos con los compañeros más similares, es decir, con aquellos que coinciden en más problemas aprobados.
    6. Ya tenemos los compañeros más coincidentes, ahora, entre esos, se buscan los menos diferentes. Se guardan los identificadores de los compañeros que coinciden en más problemas aprobados con el alumno y difieren en menos. Por ejemplo, un compañero que coincida con el alumno en 4 problemas aprobados y difiera en 2 tendrá mayor similitud que uno que coincida en 4 y difiera en 5.

7. Una vez que tenemos los identificadores de los compañeros más coincidentes y menos diferentes, obtenemos los identificadores de los problemas en los que difieren, que serán posibles recomendaciones.

8. Ahora buscamos los problemas diferentes más populares entre los compañeros, es decir, aquellos que han sido aprobados más veces por los compañeros más coincidentes y menos diferentes. Solo tenemos en cuenta los problemas aprobados ya que no tiene sentido recomendar problemas que hayan suspendido otros compañeros similares.

9. Ya podemos realizar las recomendaciones. Necesitamos recomendar tantos problemas como se nos indique por parámetro:

    1. Empezamos recomendando al alumno que repita los problemas que ha suspendido hasta el momento antes de que siga avanzando en el curso.
    2. Si no se han alcanzado el número de recomendaciones requeridas, se pasa a recomendar los problemas aprobados más repetidos por los compañeros más coincidentes y menos diferentes obtenidos en el punto 8.
    3. Si necesitamos más recomendaciones pasamos a recomendar problemas aprobados por los compañeros más coincidentes y un poco más diferentes que los menos diferentes. Es decir que, si por ejemplo, estábamos tratando con compañeros más coincidentes que diferían en 1 problema empezamos a recomendar problemas de los que difieren en más de 1.
    4. En caso de quedarnos sin problemas para recomendar, se le dará el valor *None*.

10. Una vez que tenemos las recomendaciones las insertamos en la tabla 'recommender_student' de la base de datos en *MySQL*. En caso de que existan recomendaciones previas se comprueba si son iguales y en caso de que sean diferentes se actualizan las filas de la tabla.

Llegados a este punto se renderiza la información y se muestran en la pestaña tantos problemas recomendados como se indique en el parámetro `number` de la función `get_recommendations(user_id, course_id, number)`, llamada desde el fichero *HTML* de la aplicación.

> ***Nota:*** *En caso de que el parámetro* `number` *sea mayor que el número de problemas recomendados para ese alumno en ese curso almacenados en la tabla* `recommendation_student` *de MySQL, solo se mostrarán los que haya en la tabla.*

A continuación, en las figuras de la 5-11 a la 5-28, se incluyen los diagramas de flujo de cada una de las funciones implementadas en la aplicación y una breve explicación de cada una de ellas:

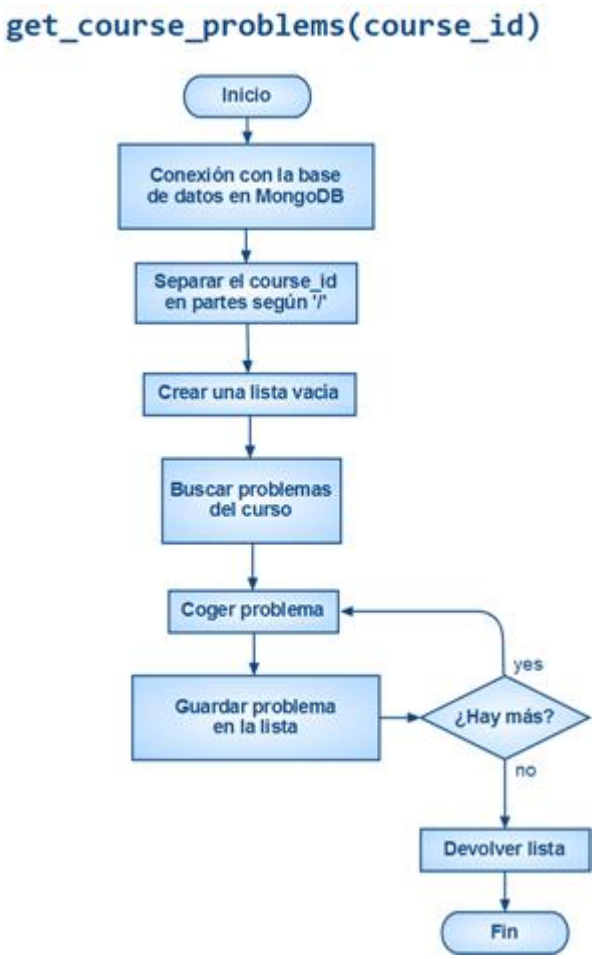


FIGURA 5-11. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_COURSE_PROBLEMS(COURSE_ID)`

La función `get_course_problems(course_id)` (ver figura 5-11) realiza una conexión con la base de datos *MongoDB* y recorre los documentos almacenados en la colección 'xmodule' buscando los módulos de tipo `problem` existentes en el curso con el identificador `course_id` pasado por parámetro. Almacena dichos módulos en una lista y al final la devuelve.

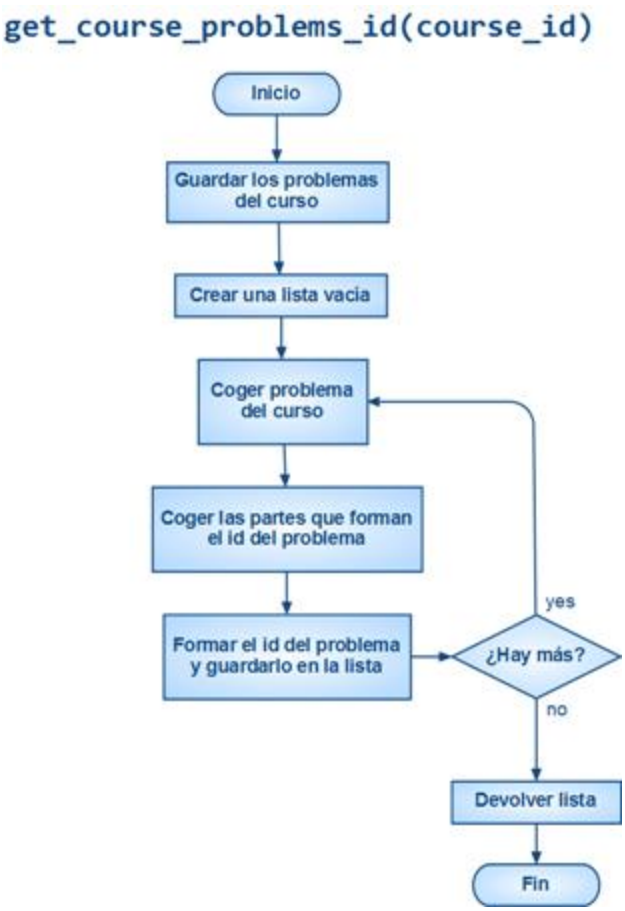


FIGURA 5-12. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_COURSE_PROBLEMS_ID(COURSE_ID)`

La función `get_course_problems_id(course_id)` (ver figura 5-12) primero obtiene los problemas del curso con identificador `course_id` mediante la función `get_course_problems(course_id)`. Posteriormente recorre la lista de problemas devuelta por la función anterior y concatena las distintas partes que forman el identificador de cada uno de ellos para generar el `module_id` (según se muestra en la tabla 5-2 del apartado 5.1.3). Estos identificadores se almacenan en una lista y al final la devuelve.

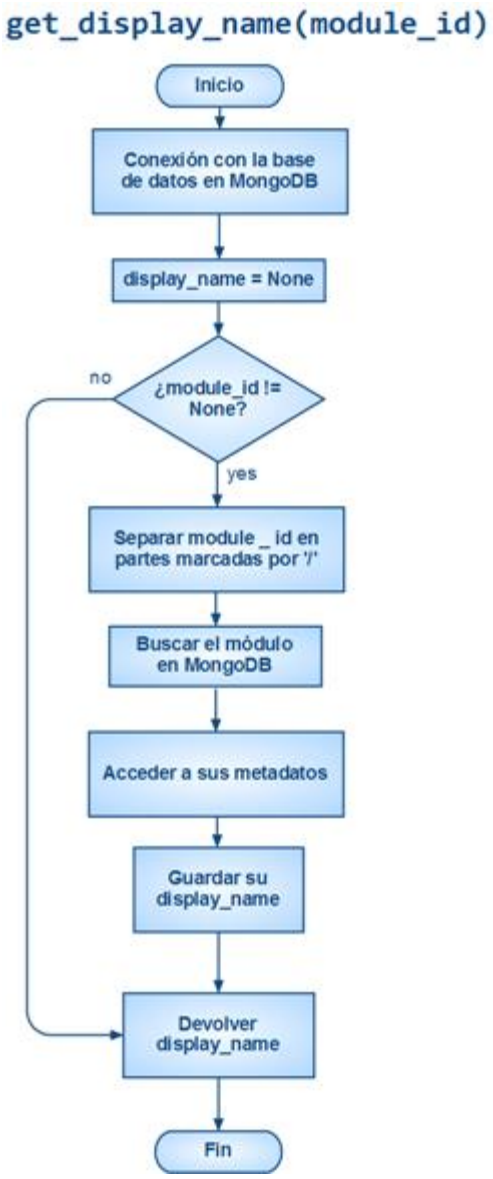


FIGURA 5-13. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_DISPLAY_NAME(MODULE_ID)`

La función `get_display_name(module_id)` (ver figura 5-13) establece una conexión con *MongoDB* y recorre los documentos de la colección 'xmodule' buscando el módulo con identificador `module_id` pasado por parámetro. Una vez que encuentra el módulo accede al campo `display_name` de sus metadatos y lo devuelve. Se utiliza para mostrar en la pestaña *Recommend Me!* el nombre asignado a ese problema por el creador del curso en lugar del `module_id` con el fin de facilitarle al alumno la búsqueda del problema dentro del curso.

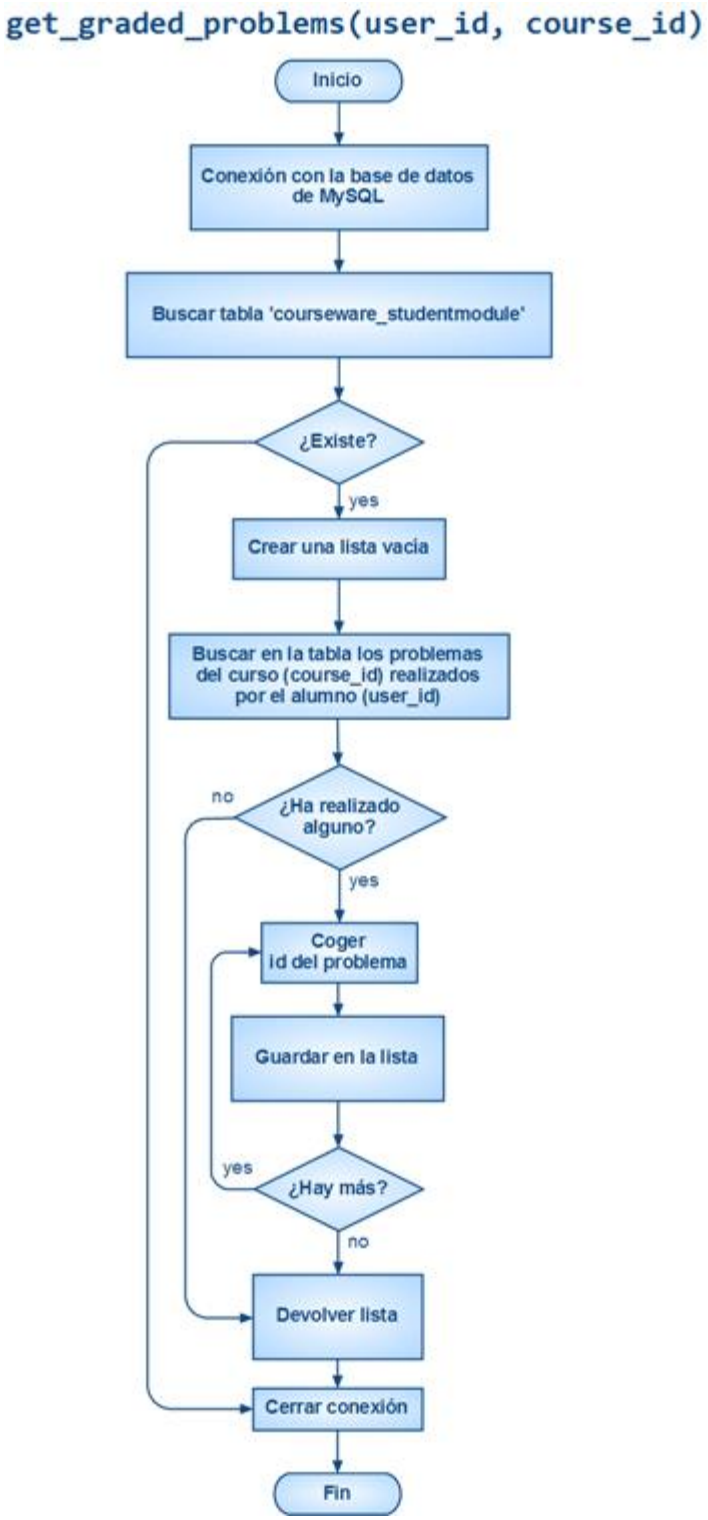


FIGURA 5-14. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_GRADED_PROBLEMS(USER_ID, COURSE_ID)`

La función `get_graded_problems(user_id, course_id)` (ver figura 5-14) establece una conexión con la base de datos *MySQL* y busca en la tabla 'courseware_studentmodule' todos los problemas realizados por el alumno con identificador `user_id` en el curso con identificador `course_id`. Los identificadores de dichos problemas se almacenan en una lista y se devuelve.

La función `get_ids(user_id, course_id)` (ver figura 5-15) establece una conexión con la base de datos *MySQL* y busca en la tabla 'courseware_studentmodule' todos los identificadores de los alumnos compañeros del alumno con identificador `user_id` en el curso con identificador `course_id`. Los identificadores de los compañeros se almacenan en una lista y se devuelve.

Las funciones `get_passed_problems(user_id, course_id)` y `get_failed_problems(user_id, course_id)` (ver figuras 5-16 y 5-17) se encargan de obtener los problemas aprobados y suspensos, respectivamente, del alumno con identificador `user_id` en el curso con identificador `course_id`. Para ello:

- La función `get_passed_problems(user_id, course_id)` primero obtiene los identificadores de los problemas realizados por el alumno mediante la función `get_graded_problems(user_id, course_id)`, establece una conexión con la base de datos *MySQL* y accede a la tabla 'courseware_studentmodule' para buscar la nota obtenida por el alumno y la nota máxima que se puede obtener en cada uno de ellos. Si la nota de un problema es igual o superior a la mitad de la nota máxima que se puede obtener para dicho problema guarda su identificador en una lista que es devuelta al final.

- La función `get_failed_problems(user_id, course_id)` obtiene los problemas realizados por el alumno con identificador `user_id` en el curso con identificador `course_id` mediante la función `get_graded_problems(user_id, course_id)` y sus problemas aprobados mediante la función `get_passed_problems(user_id, course_id)` y almacena en una lista que devuelve al final los identificadores de los problemas que hayan sido realizados pero no aprobados, es decir, suspensos.

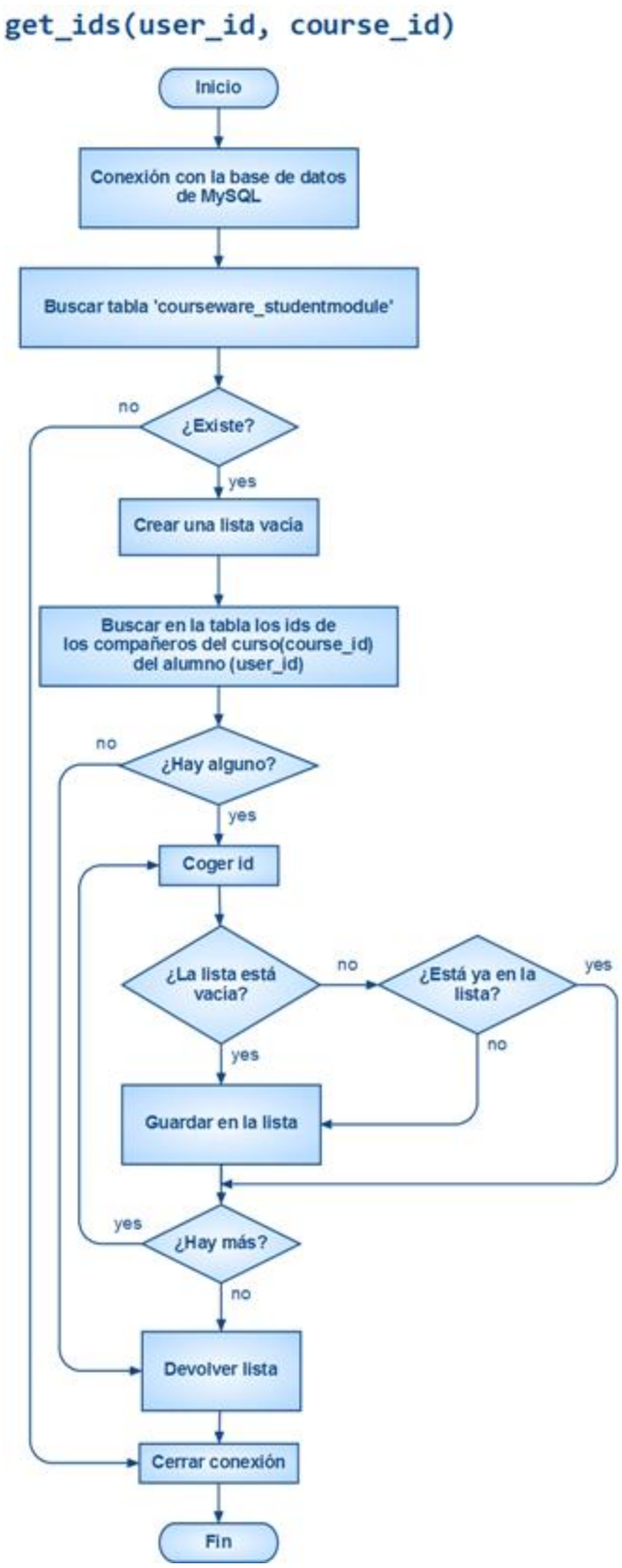


FIGURA 5-15. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_IDS(USER_ID, COURSE_ID)`

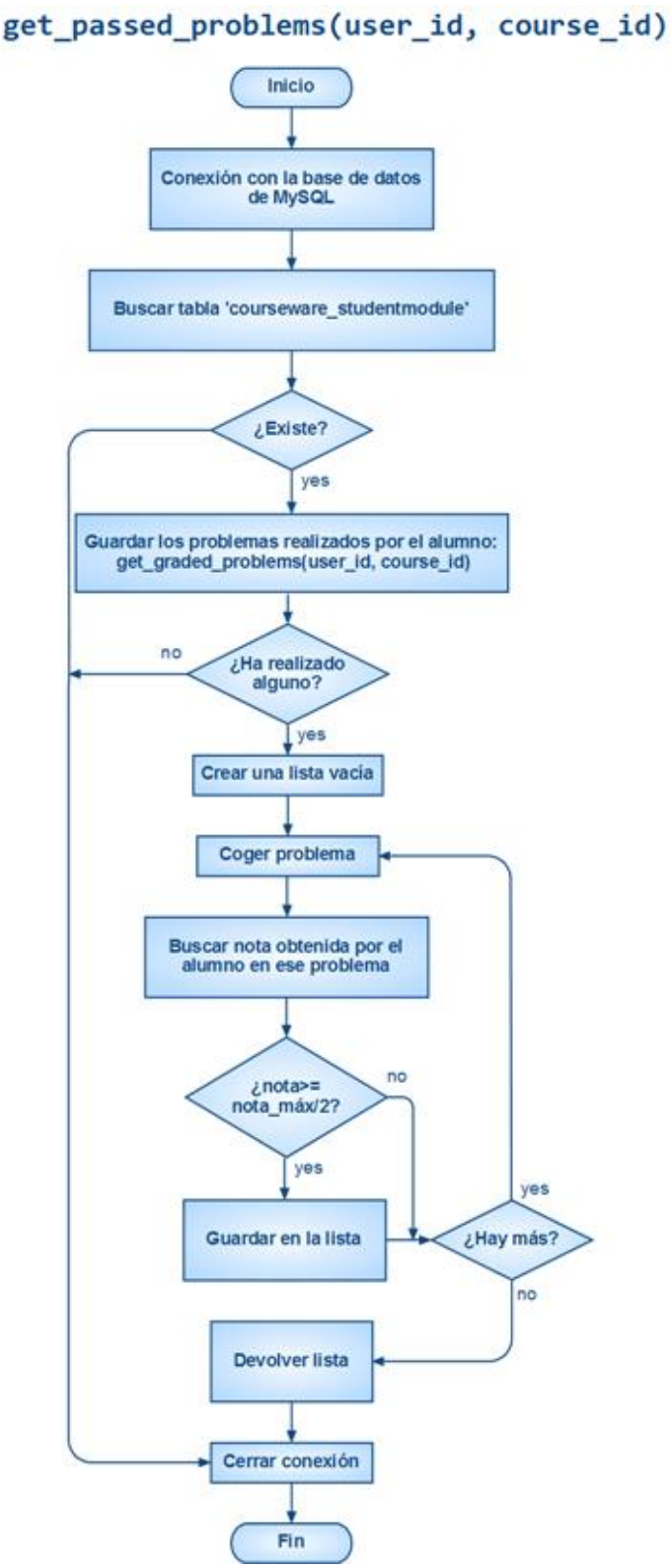


FIGURA 5-16. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_PASSED_PROBLEMS(USER_ID, COURSE_ID)`

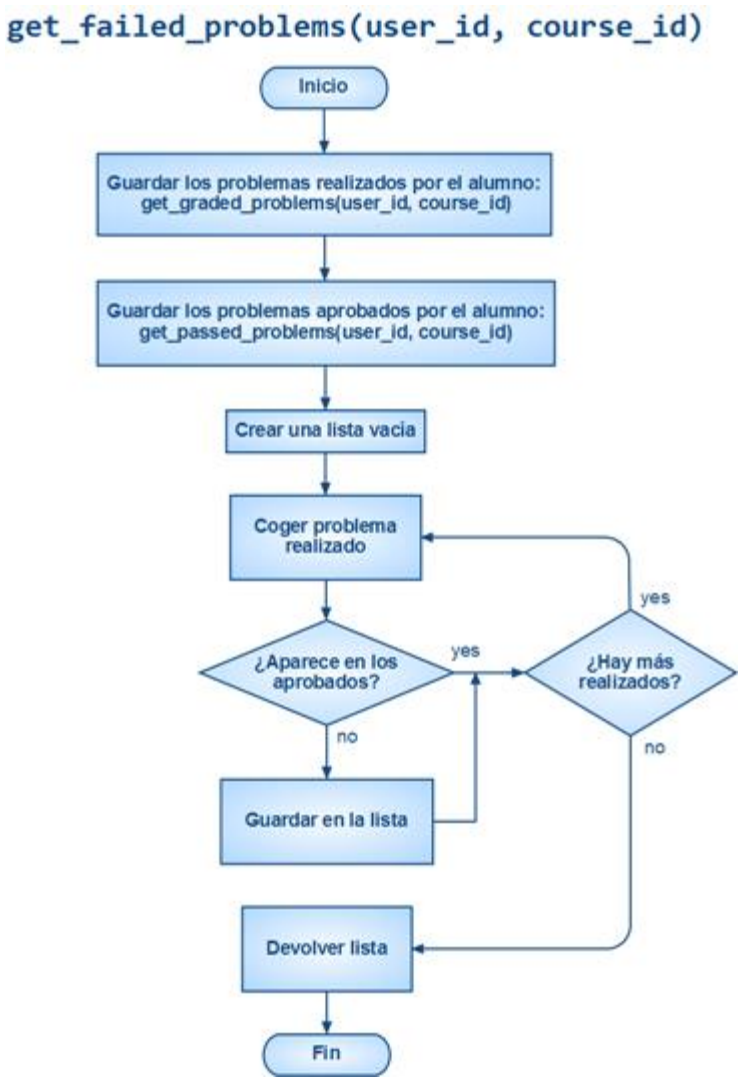


FIGURA 5-17. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_FAILED_PROBLEMS (USER_ID, COURSE_ID)`

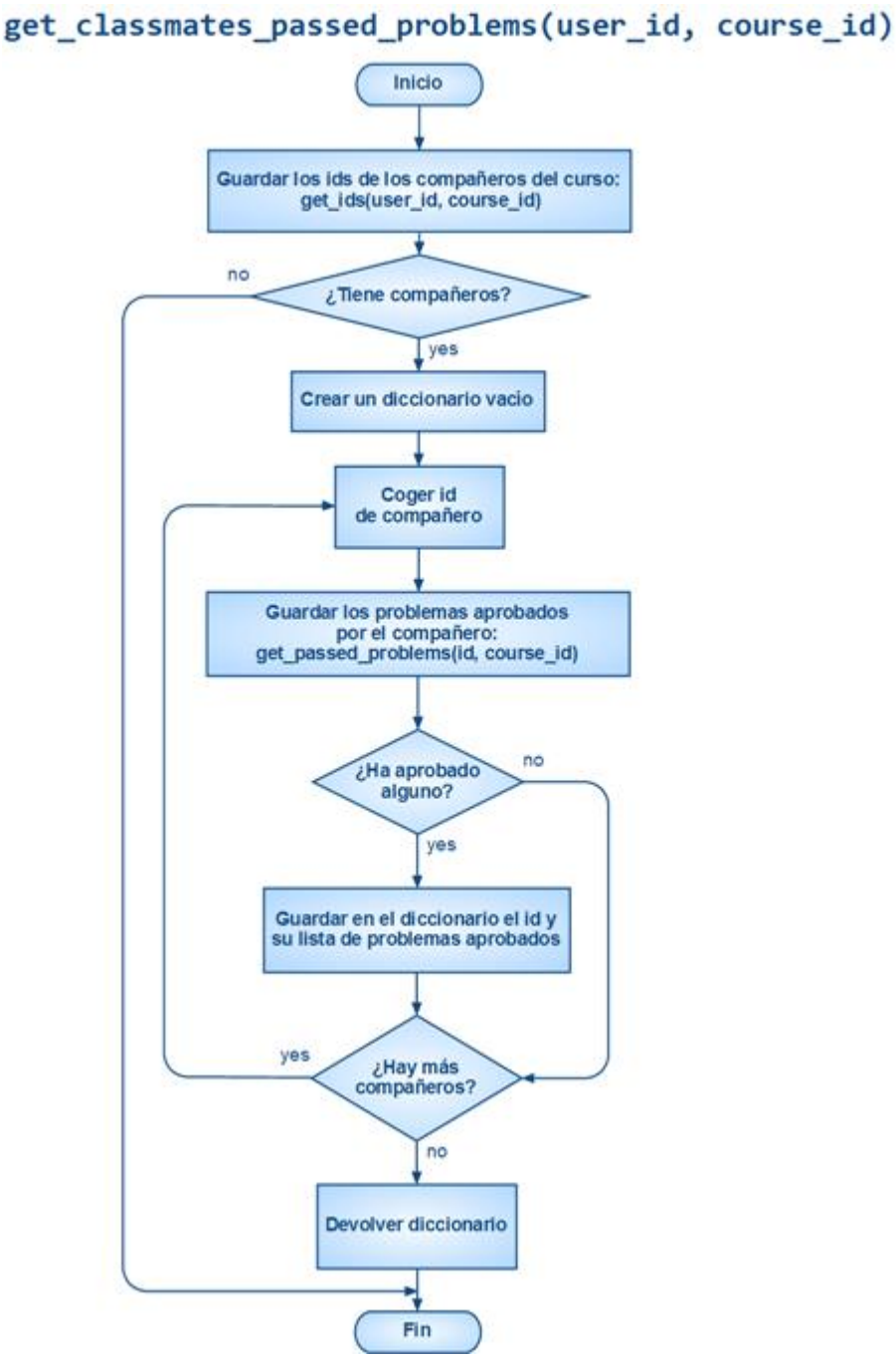


FIGURA 5-18. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_CLASSMATES_PASSED_PROBLEMS(USER_ID, COURSE_ID)`

La función `get_classmates_passed_problems(user_id, course_id)` (ver figura 5-18) obtiene los identificadores de los alumnos compañeros del alumno con identificador `user_id` en el curso con identificador `course_id` mediante la función `get_ids(user_id, course_id)`. Posteriormente busca los problemas aprobados por cada compañero y los almacena en una lista. El identificador de cada compañero y su lista de problemas aprobados conforman el diccionario que será devuelto al final: {identificador del compañero: {[lista de problemas aprobados]}.

La función `get_number_of_passed_coincidences(user_id, course_id)` (ver figura 5-19) se encarga de calcular el número de problemas aprobados en los que coinciden el alumno con identificador `user_id` con cada uno de sus compañeros del curso con identificador `course_id`. Se almacena en un diccionario, que se devuelve al final, el identificador de cada compañero y el número de problemas aprobados en los que coincide con el usuario de la siguiente forma: {identificador de compañero: número de problemas aprobados en los que coinciden}.

Por otro lado, la función `get_passed_coincidences(user_id, course_id)` (ver figura 5-20) devuelve un diccionario en el que se almacenan los identificadores de los compañeros y los problemas aprobados en los que coinciden con el alumno con identificador `user:_id` de la siguiente forma: {identificador del compañero: [lista de problemas aprobados en los que coincide con el alumno]}. Para ello se obtiene la lista de problemas aprobados por el alumno con identificador `user_id` mediante la función `get_passed_problems(user_id, course_id)` y el diccionario en el que se almacenan los problemas aprobados por cada uno de sus compañeros de curso mediante la función `get_classmates_passed_problems(user_id, course_id)` y se comparan para encontrar los problemas en los que coinciden.

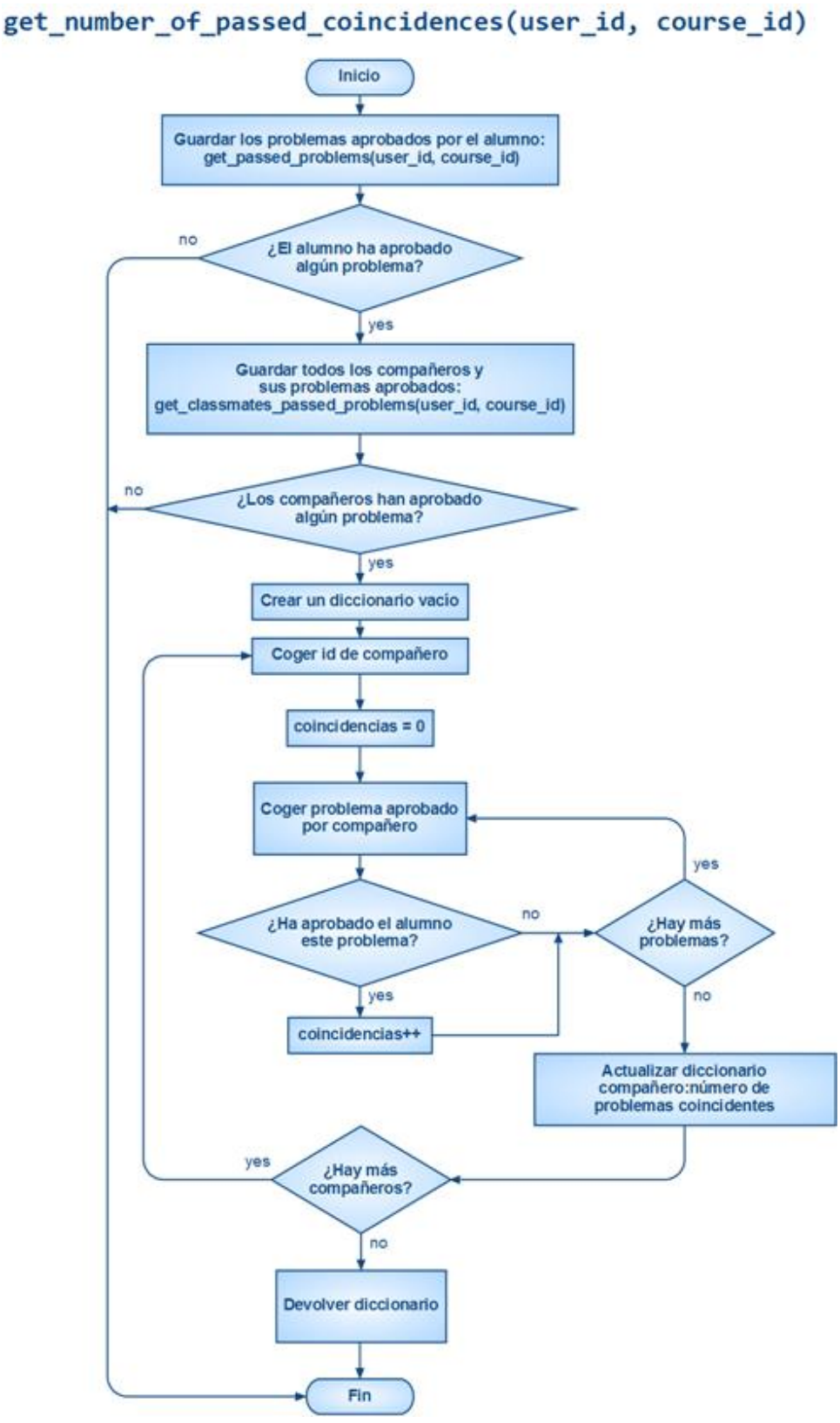


FIGURA 5-19. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_NUMBER_OF_PASSED_COINCIDENCES(USER_ID, COURSE_ID)`

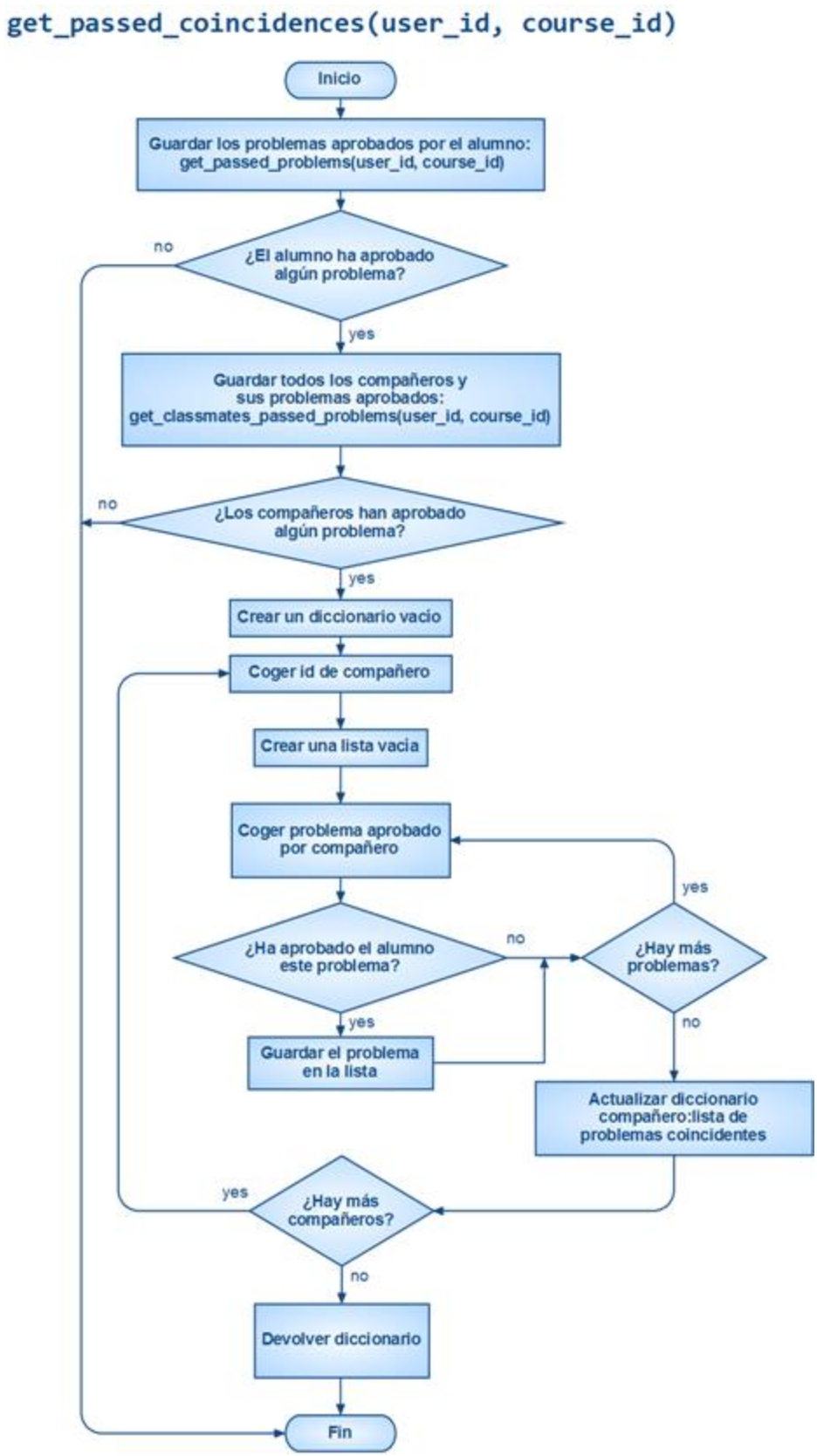


FIGURA 5-20. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_PASSED_COINCIDENCES(USER_ID, COURSE_ID)`

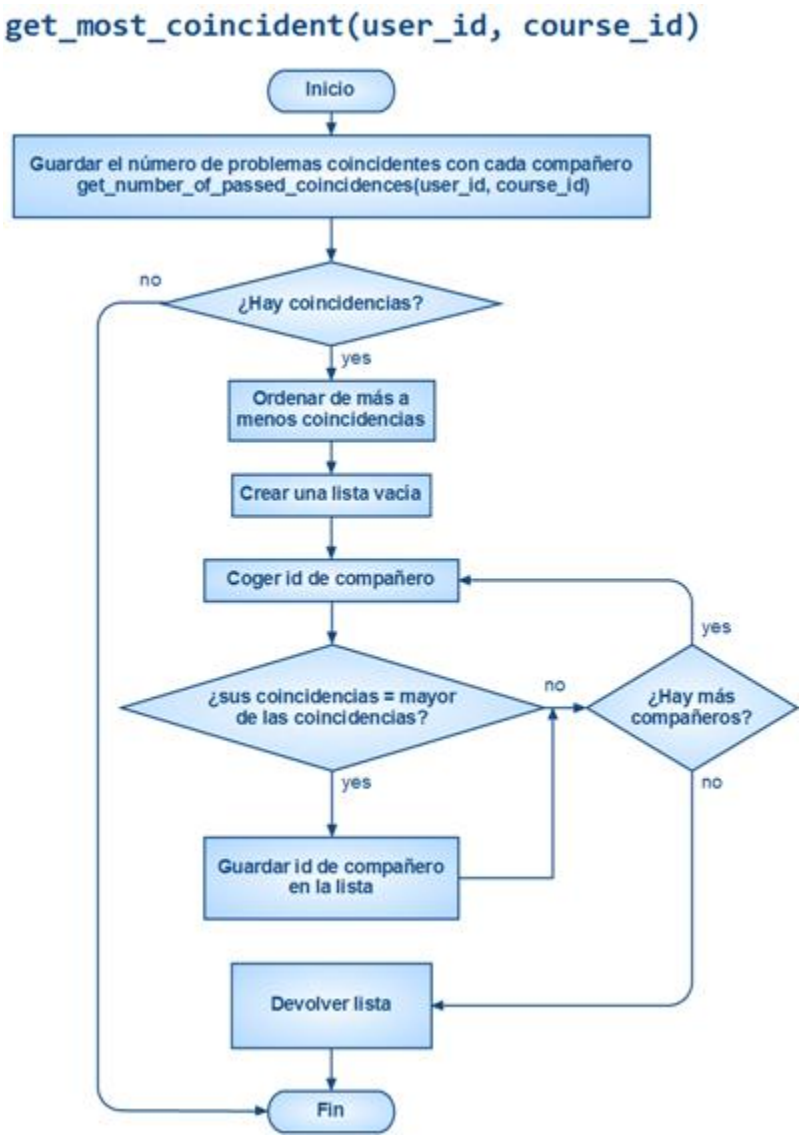


FIGURA 5-21. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_MOST_COINCIDENT(USER_ID, COURSE_ID)`

La función `get_most_coincident(user_id, course_id)` (ver figura 5-21) se encarga de buscar los compañeros del alumno con identificador `user_id` que tienen mayor número de problemas aprobados en común obteniéndolos del diccionario devuelto por la función `get_number_of_passed_coincidences(user_id, course_id)`. Los identificadores de los compañeros se guardan en una lista que se devuelve al final.

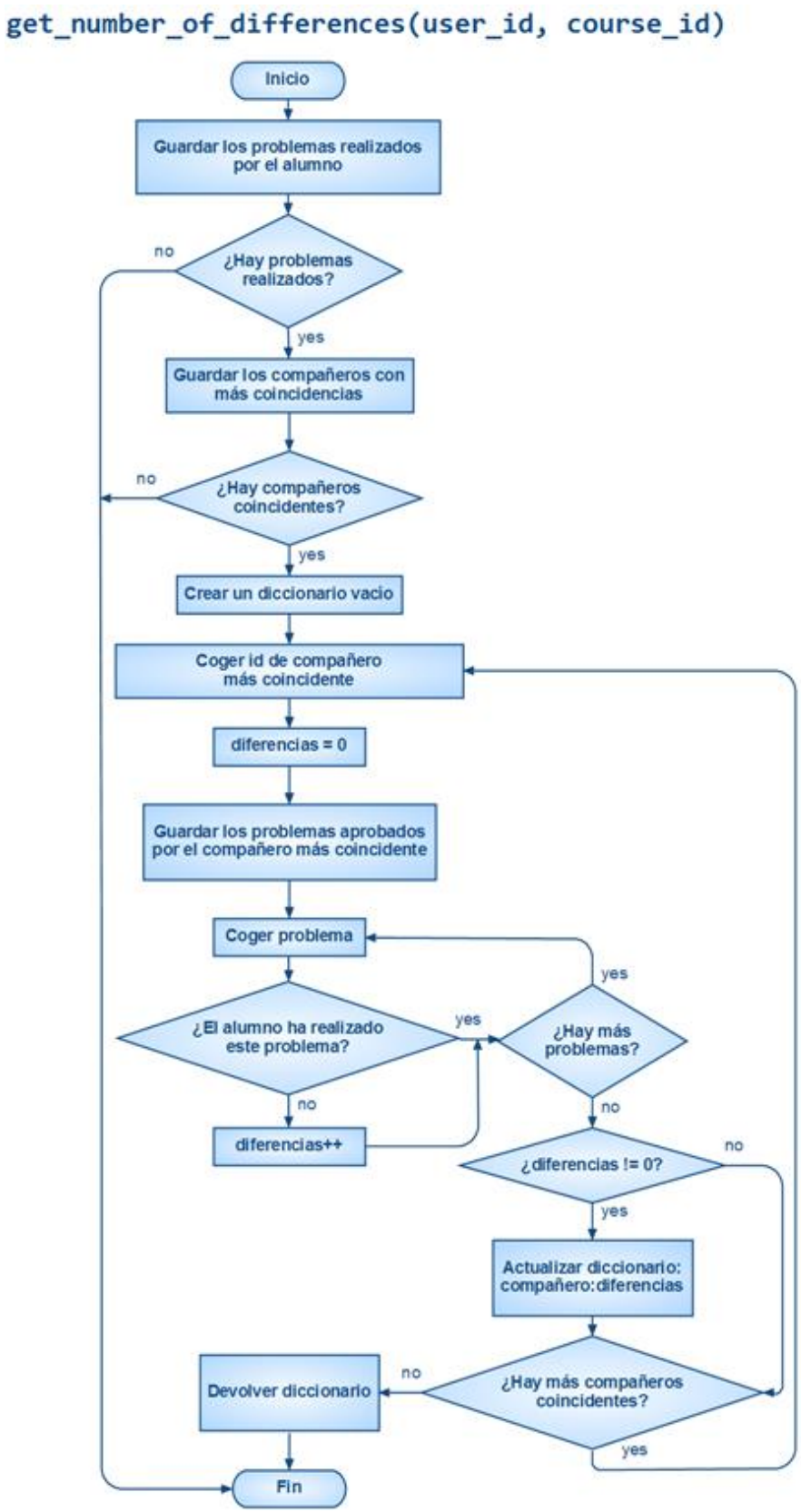


FIGURA 5-22. DIAGRAMA DE FLUJO DE LA FUNCIÓN GET_NUMBER_OF_DIFFERENCES(USER_ID, COURSE_ID)

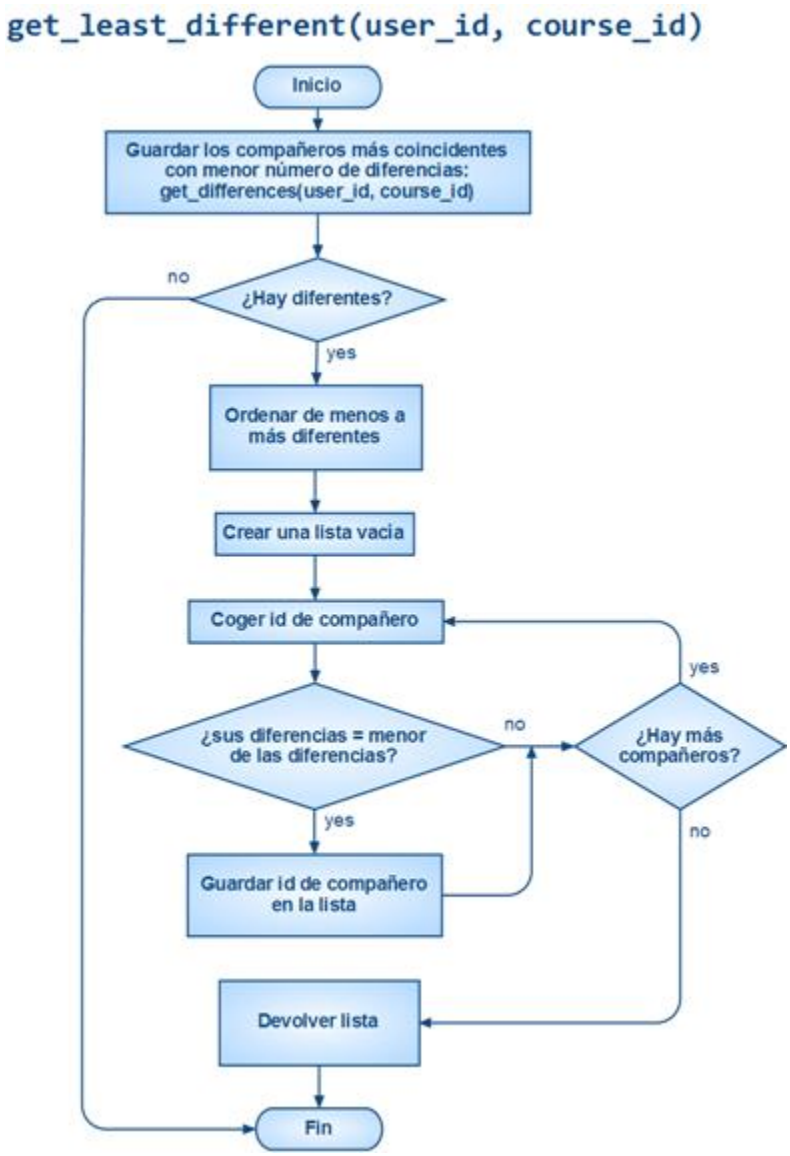


FIGURA 5-23. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_LEAST_DIFFERENT(USER_ID, COURSE_ID)`

La función `get_number_of_differences(user_id, course_id)` (ver figura 5-22) se encarga de calcular el número de problemas aprobados en los que difiere cada compañero más coincidente con el alumno con identificador `user_id`. Se devuelve un diccionario formado por: {identificador de compañero más coincidente: número de problemas aprobados diferentes}. La función `get_least_different(user_id, course_id)` (ver figura 5-23) se encarga de seleccionar los compañeros más coincidentes y menos diferentes y almacenar su identificador en la lista que se devuelve.

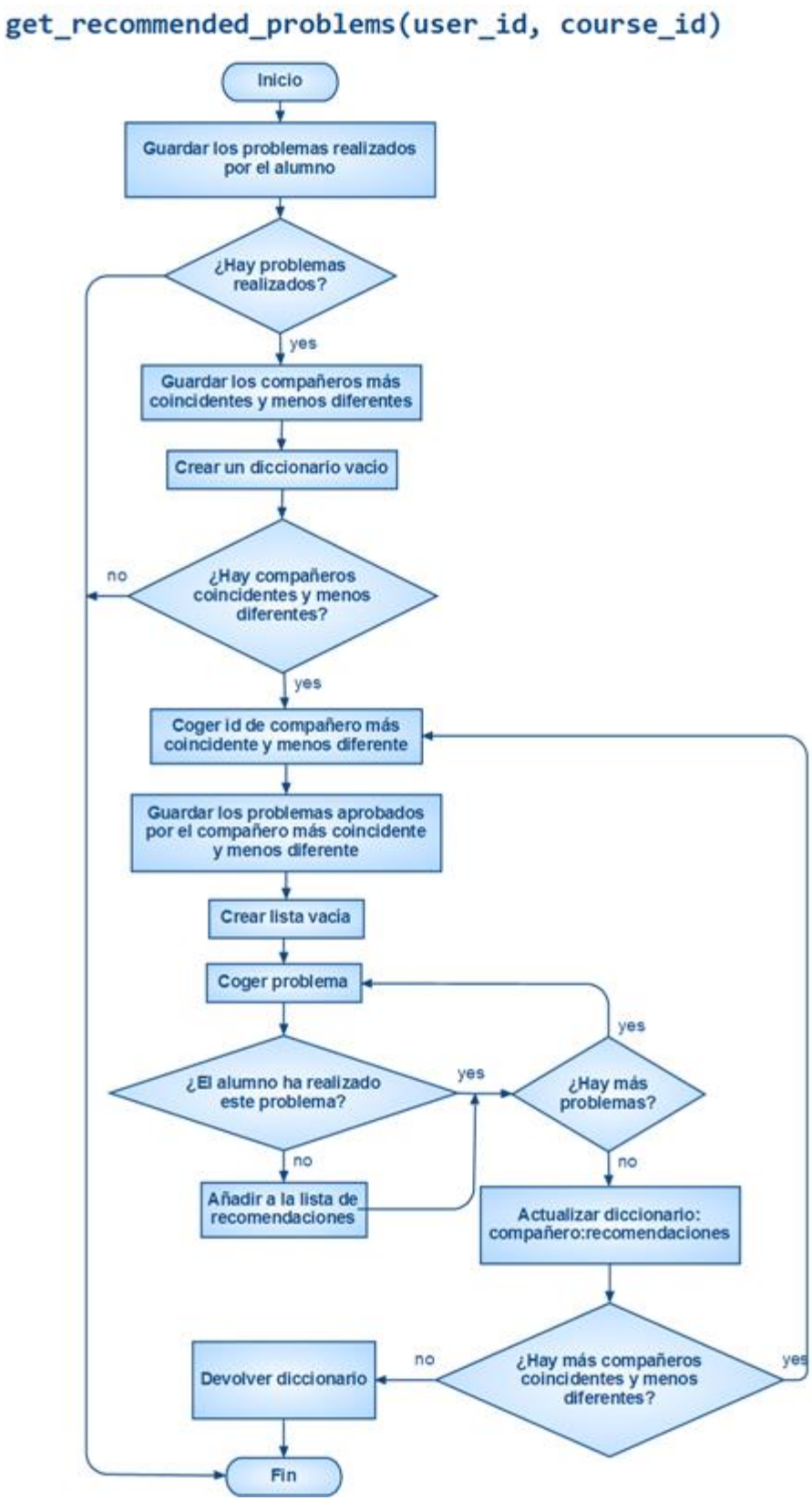


FIGURA 5-24. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_RECOMMENDED_PROBLEMS(USER_ID, COURSE_ID)`

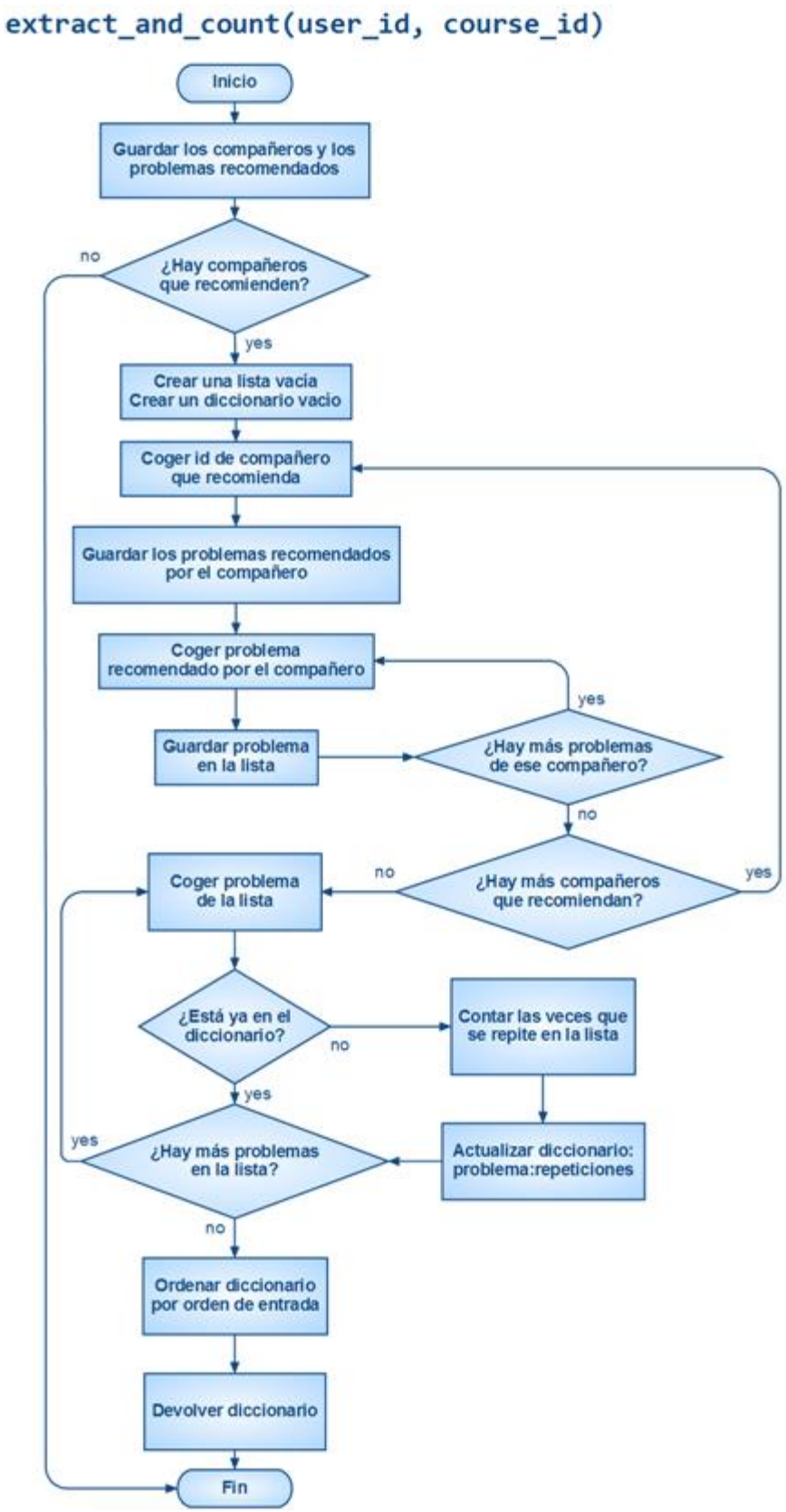


FIGURA 5-25. DIAGRAMA DE FLUJO DE LA FUNCIÓN `EXTRACT_AND_COUNT(USER_ID, COURSE_ID)`

La función `get_recommended_problems(user_id, course_id)` (ver figura 5-24) se encarga de seleccionar aquellos problemas susceptibles de ser recomendados al alumno con identificador `user_id`, es decir, aquellos problemas aprobados por sus compañeros más coincidentes y menos diferentes y que éste todavía no ha realizado.

Nos referimos a esos problemas como 'problemas susceptibles de ser recomendados' porque no se recomendarán todos, sino solo aquellos más repetidos, es decir, un problema que se haya realizado y aprobado ocho veces será más susceptible de ser recomendado que uno que solo se haya realizado y aprobado en dos ocasiones. La función `extract_and_count(user_id, course_id)` (ver figura 5-25) se encarga de obtener los problemas susceptibles de ser recomendados y contar el número de veces que se repite cada uno. Devuelve un diccionario formado por: {identificador de problema susceptible de ser recomendado: número de veces que se ha realizado y aprobado}.

Pero no solo se recomiendan problemas por parte de los compañeros más coincidentes y menos diferentes. La función `get_best_recommendations(user_id, course_id, number)` (ver figura 5-26) devuelve una lista con tantas recomendaciones como las indicadas por el parámetro `number` según lo siguiente:

- Primero se recomiendan los problemas suspensos del alumno con identificador `user_id`.
- Si todavía no se ha cubierto el número de recomendaciones necesarias se recomiendan los problemas aprobados más repetidos por los compañeros más coincidentes y menos diferentes.
- Si todavía no se ha cubierto el número de recomendaciones necesarias se recomiendan problemas de otros compañeros más coincidentes y menos diferentes, es decir, se excluyen los compañeros más coincidentes y menos diferentes de los que ya se han tomado problemas susceptibles de ser recomendados. Por ejemplo, si ya se han recomendado los problemas de aquellos alumnos coincidentes que difieren en un problema se pasará a recomendar problemas de los alumnos coincidentes que difieran en dos.

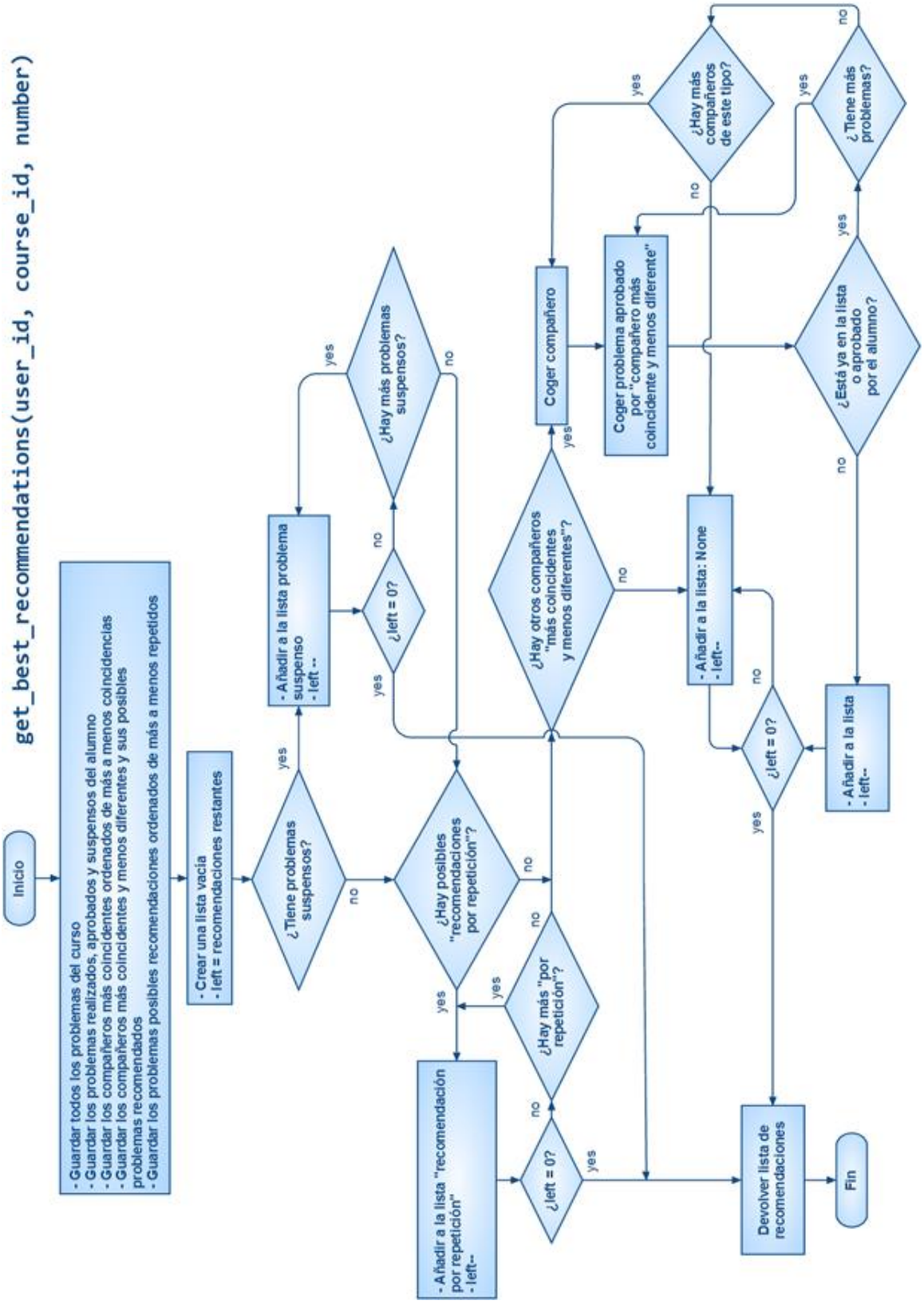


FIGURA 5-26. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_BEST_RECOMMENDATIONS(USER_ID, COURSE_ID, NUMBER)`

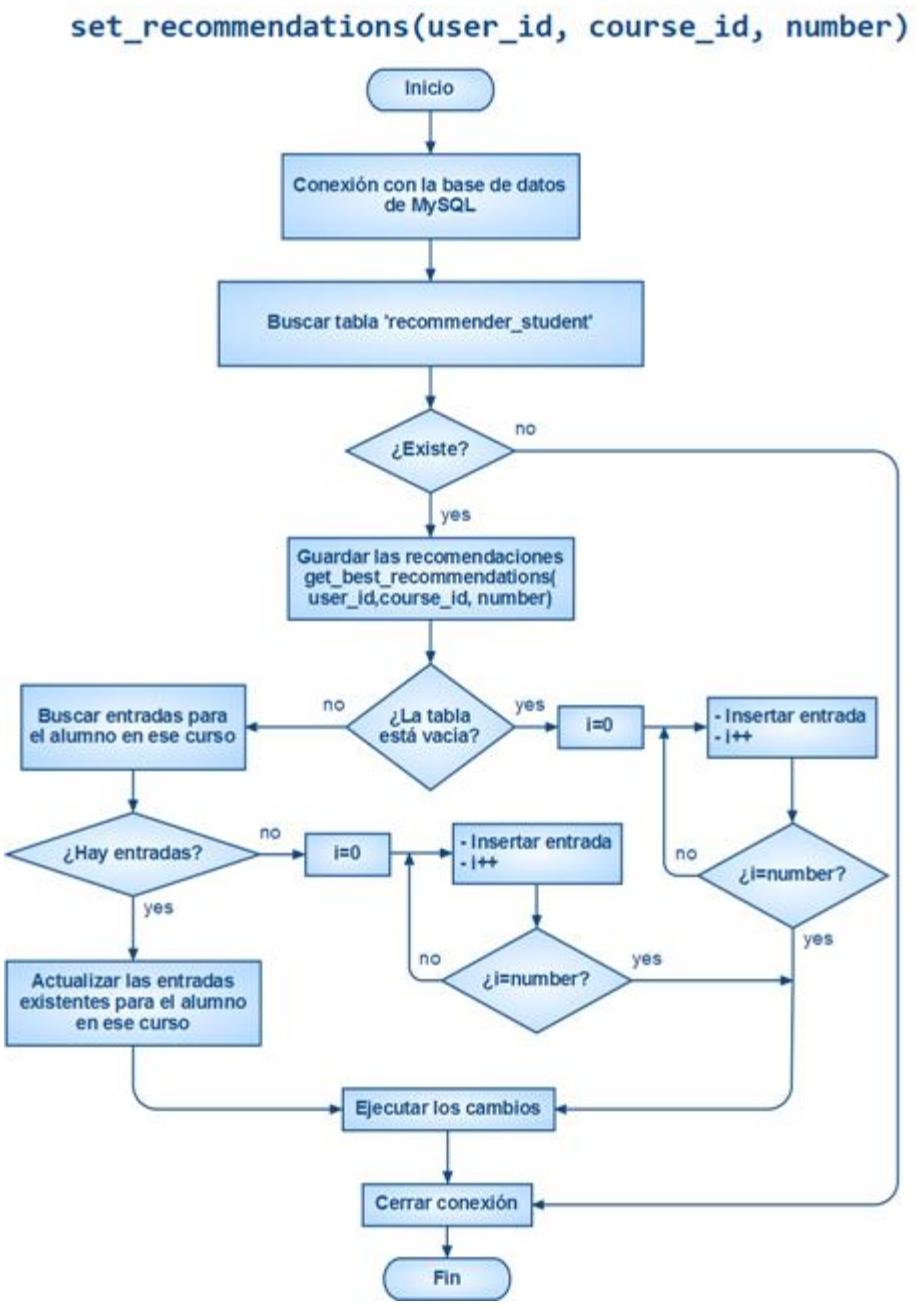


FIGURA 5-27. DIAGRAMA DE FLUJO DE LA FUNCIÓN SET_RECOMMENDATIONS(USER_ID, COURSE_ID, NUMBER)

La función `set_recommendations(user_id, course_id, number)` (ver figura 5-27) establece una conexión con la base de datos MySQL y almacena en la tabla 'recommender_student' las recomendaciones para el alumno con identificador `user_id` en el curso con identificador `course_id` indicadas por el parámetro `number`.

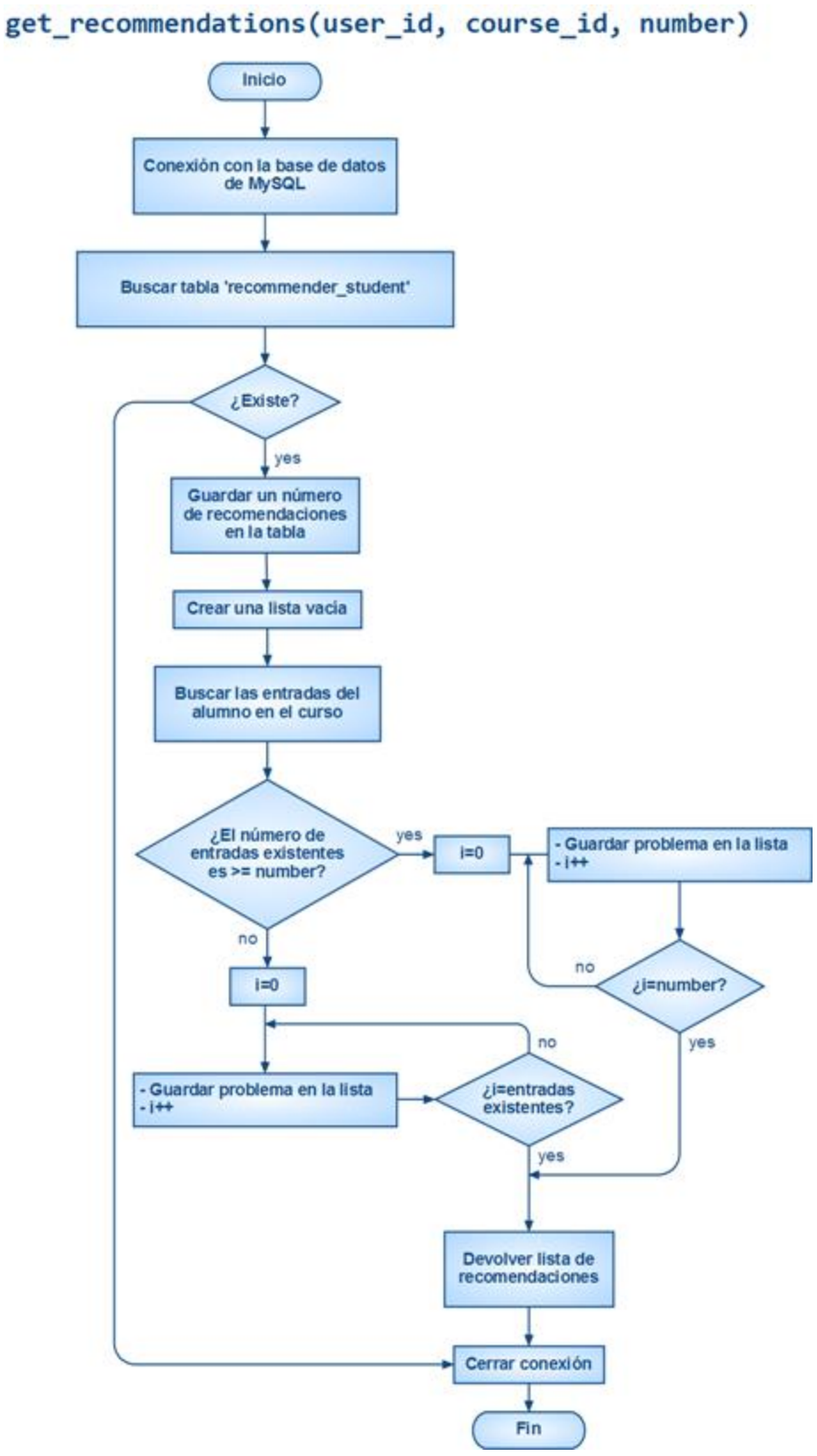


FIGURA 5-28. DIAGRAMA DE FLUJO DE LA FUNCIÓN `GET_RECOMMENDATIONS(USER_ID, COURSE_ID, NUMBER)`

La función `get_recommendations(user_id, course_id, number)` (ver figura 5-28) establece una conexión con la base de datos *MySQL,* accede a la tabla 'recommender student' y devuelve el número de recomendaciones para el alumno con identificador `user_id` (en ese punto del curso) determinadas por el parámetro `number`.

6. Pruebas y Resultados

En este apartado se desarrollarán, en detalle, algunos casos relevantes de prueba y se analizarán los resultados con el objetivo de demostrar que se han cubierto todas las posibilidades.

6.1. Caso 1: Único alumno registrado en el curso. Todavía no ha realizado ningún problema

En caso de ser el único alumno registrado en el curso y todavía no haber realizado ningún problema no se pueden recomendar problemas de sus compañeros ni problemas que haya realizado y suspendido, por lo tanto, el alumno ve el mensaje de la imagen 6-1 en la pestaña *Recommend Me!*

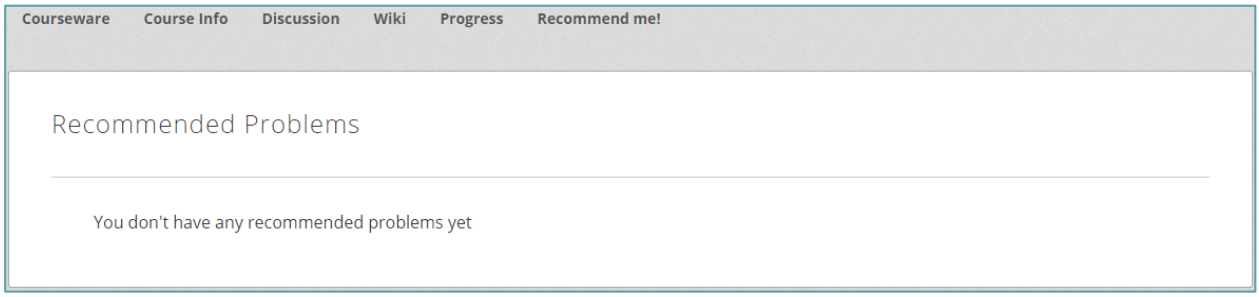


FIGURA 6-1. MENSAJE DE AVISO PARA EL ALUMNO EN EL CASO 1

En la tabla 'recommender_student' se crean tantas entradas por alumno y curso como indique el parámetro `number`. En este caso cuatro nuevas filas cuyo campo `module_id` tiene valor *NULL*. Esto lo podemos ver en la imagen 6-2.

```
mysql> select * from recommender_student;
+----+---------+-------------------+-----------+
| id | user_id | course_id         | module_id |
+----+---------+-------------------+-----------+
| 61 |       2 | uc3m/CP01/2014_T1 | NULL      |
| 62 |       2 | uc3m/CP01/2014_T1 | NULL      |
| 63 |       2 | uc3m/CP01/2014_T1 | NULL      |
| 64 |       2 | uc3m/CP01/2014_T1 | NULL      |
+----+---------+-------------------+-----------+
4 rows in set (0.00 sec)
```

FIGURA 6-2. ESTADO DE LA TABLA RECOMMENDER_STUDENT EN EL CASO 1

En consola obtenemos la información mostrada en la imagen 6-3:

```
------------------------------------------------------------------------------------------------------------------------
                                                    Logged in student:  2
                                                    Recommendations for the course: uc3m/CP01/2014_T1
                                                    Number of recommendations needed: 3
------------------------------------------------------------------------------------------------------------------------

Student PASSED and FAILED problems:

The student doesn't have any graded problems yet. The student has to start the course to get any recommendations

------------------------------------------------------------------------------------------------------------------------
```

FIGURA 6-3. MENSAJES EN CONSOLA EN EL CASO 1

Arriba a la derecha se muestra el `user_id`, el `course_id` y el número de recomendaciones que queremos que se le muestren al estudiante en la pestaña *Recommend Me!*. En este caso, como se indica en el mensaje de consola no se ha podido hacer ninguna recomendación al alumno, ya que para que se le pueda hacer alguna recomendación primero debe empezar el curso.

6.2. Caso 2: Único alumno registrado. Ha realizado varios problemas pero ha aprobado todos

En caso de ser el único alumno registrado en el curso y haber realizado algún problema, el recomendador buscará, en primer lugar, los problemas suspensos del alumno. En este caso ha aprobado todos, luego no se podrá realizar ninguna recomendación. El alumno vuelve a obtener el mensaje de la imagen 6-1.

La tabla `recommender_student` no experimenta ningún cambio. Se mantiene en el mismo estado que en la figura 6-2.

En consola obtenemos la información mostrada en la imagen 6-4:

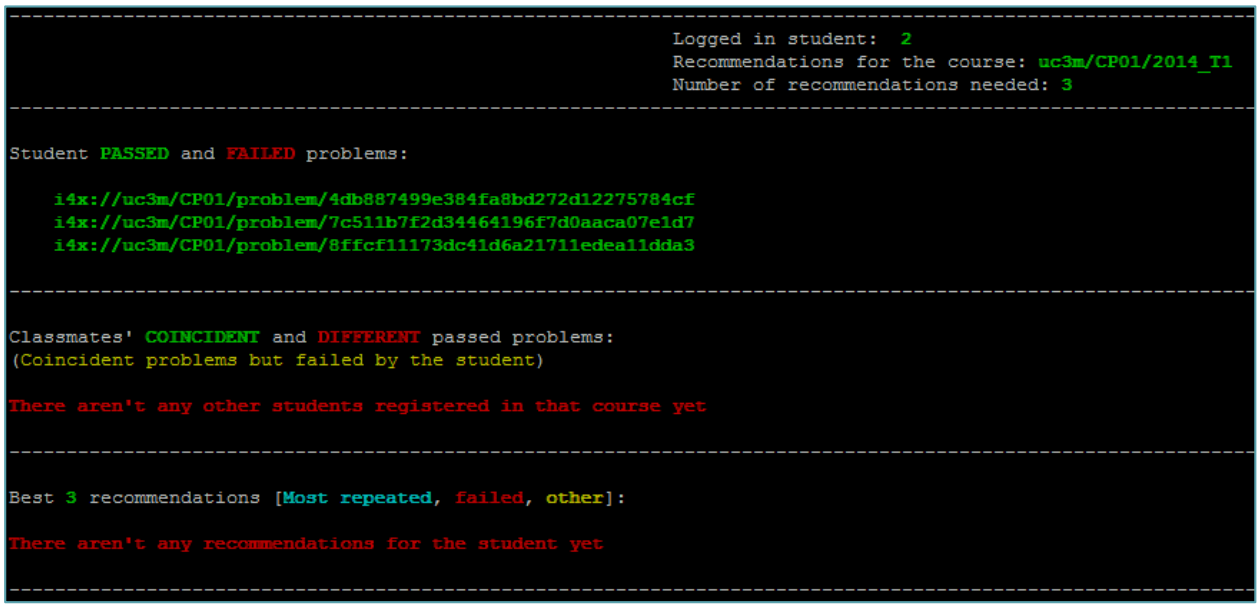


FIGURA 6-4. MENSAJES EN CONSOLA EN EL CASO 2

Observamos los problemas aprobados por el alumno en color verde. Como no tiene ninguno suspenso ni tiene más compañeros no hay recomendaciones.

6.3. Caso 3: Único alumno registrado que ha realizado varios problemas y suspendido alguno

En caso de ser el único alumno registrado en el curso y haber realizado algunos problemas, la recomendación se hará solo en base a los problemas que haya realizado y suspendido. La imagen 6-5 muestra la recomendación para el alumno:

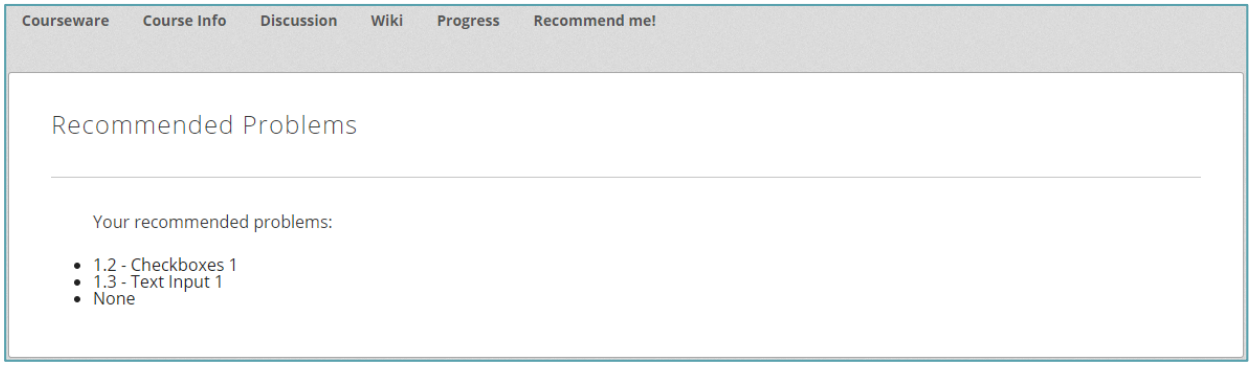


FIGURA 6-5. PROBLEMAS RECOMENDADOS PARA EL ALUMNO EN EL CASO 3

Podemos observar que se le han recomendado dos problemas en lugar de tres, esto ocurre porque el alumno solo ha suspendido dos problemas hasta el momento, por lo que solo se le pueden hacer dos recomendaciones.

En la imagen 6-6 se representa la tabla con las entradas para el alumno. El valor del campo `module_id` se corresponde con el nombre de los problemas de la imagen 6-5.

```
mysql> select * from recommender_student;
+----+---------+------------------+-----------------------------------------------------------------+
| id | user_id | course_id        | module_id                                                       |
+----+---------+------------------+-----------------------------------------------------------------+
| 73 |       2 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e |
| 74 |       2 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337 |
| 75 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
| 76 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
+----+---------+------------------+-----------------------------------------------------------------+
4 rows in set (0.00 sec)
```

FIGURA 6-6. ESTADO DE LA TABLA RECOMMENDER_STUDENT EN EL CASO 3

En la consola obtenemos la información mostrada en la imagen 6-7. Podemos ver en rojo los problemas que ha realizado el alumno y ha suspendido hasta el momento, por lo que serán los problemas recomendados. Como ha suspendido menos problemas de los que se necesitan recomendar solo se muestra el número de problemas disponibles para recomendar, en este caso dos.

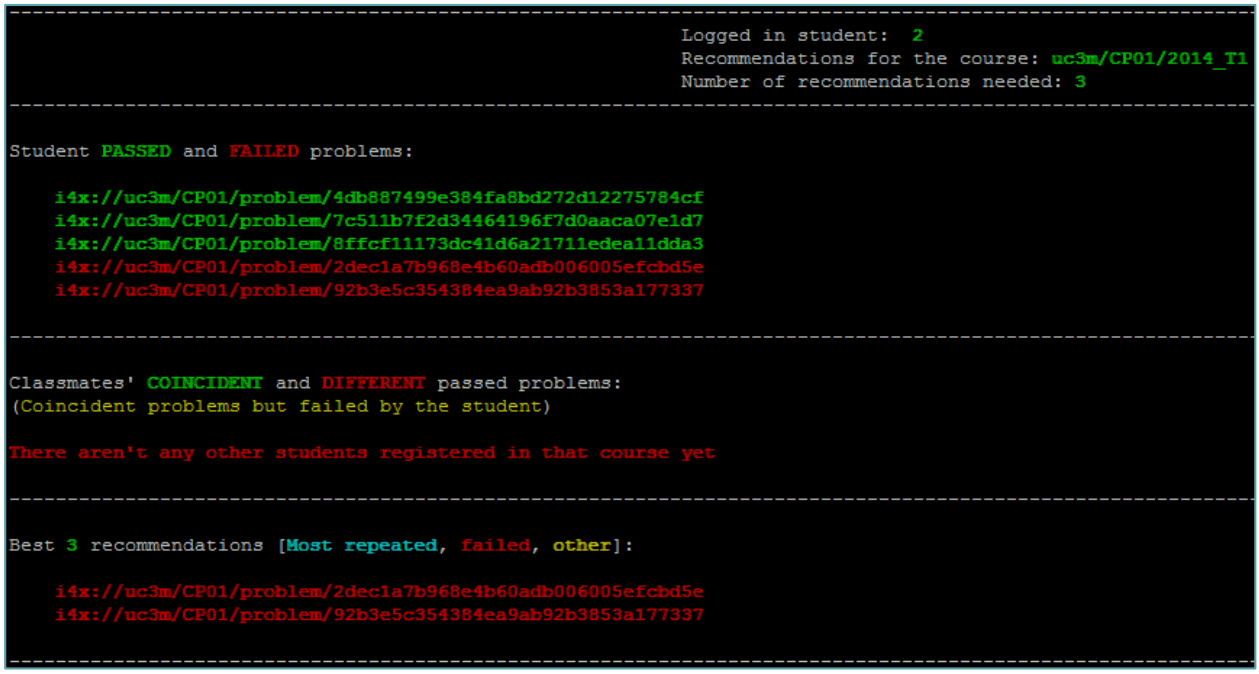


FIGURA 6-7. MENSAJES EN CONSOLA EN EL CASO 3

6.4. Caso 4: Varios alumnos registrados. El actual más avanzado

En este caso el alumno ha repetido los problemas que había suspendido y que le fueron recomendados en el Caso 3 y ha continuado con el curso. Otros alumnos se han registrado y han empezado el curso.

En la imagen 6-8 vemos las nuevas recomendaciones para el alumno.

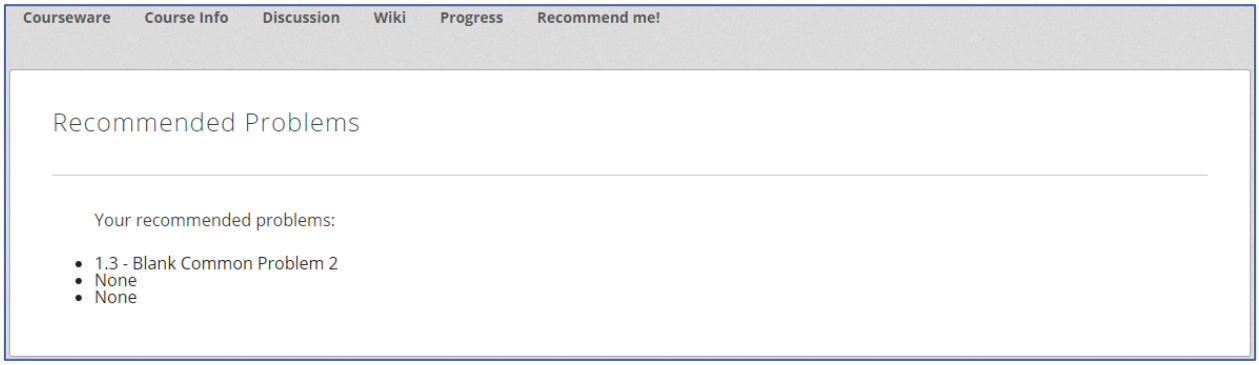


FIGURA 6-8. PROBLEMAS RECOMENDADOS PARA EL ALUMNO EN EL CASO 4

En la figura 6-9 comprobamos que se ha modificado la tabla y se ha actualizado con la nueva recomendación:

```
mysql> select * from recommender_student;
+----+---------+------------------+-----------------------------------------------------------+
| id | user_id | course_id        | module_id                                                 |
+----+---------+------------------+-----------------------------------------------------------+
| 73 |       2 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/bfe51466cf0746bbad961f7a00dcf7b7 |
| 74 |       2 | uc3m/CP01/2014_T1 | NULL                                                      |
| 75 |       2 | uc3m/CP01/2014_T1 | NULL                                                      |
| 76 |       2 | uc3m/CP01/2014_T1 | NULL                                                      |
+----+---------+------------------+-----------------------------------------------------------+
4 rows in set (0.00 sec)
```

FIGURA 6-9. ESTADO DE LA TABLA RECOMMENDER_STUDENT EN EL CASO 4

A continuación, observando la imagen 6-10, detallamos los motivos por los que el alumno solo tiene un problema recomendado a pesar de que ya hay más alumnos registrados en el curso:

- Vemos que el alumno ha repetido y aprobado los problemas recomendados en el Caso 3.
- Ha continuado el curso y ha suspendido un problema hasta ese momento.
- Se muestran los problemas en los que coincide y difiere con cada uno de sus compañeros. En este caso, como hemos dicho, el alumno va más adelantado que el resto por lo que todos los problemas realizados por sus compañeros ya los ha realizado y aprobado, no difiere en ninguno.

- Una vez obtenidos los compañeros más similares (el 9 coincide en cinco problemas aprobados), el algoritmo busca los menos diferentes, pero como en este caso no han realizado problemas distintos a los realizados por el alumno se muestra un mensaje de aviso.
- Solo tiene una recomendación, que se corresponde con el problema que el alumno lleva suspenso hasta el momento.

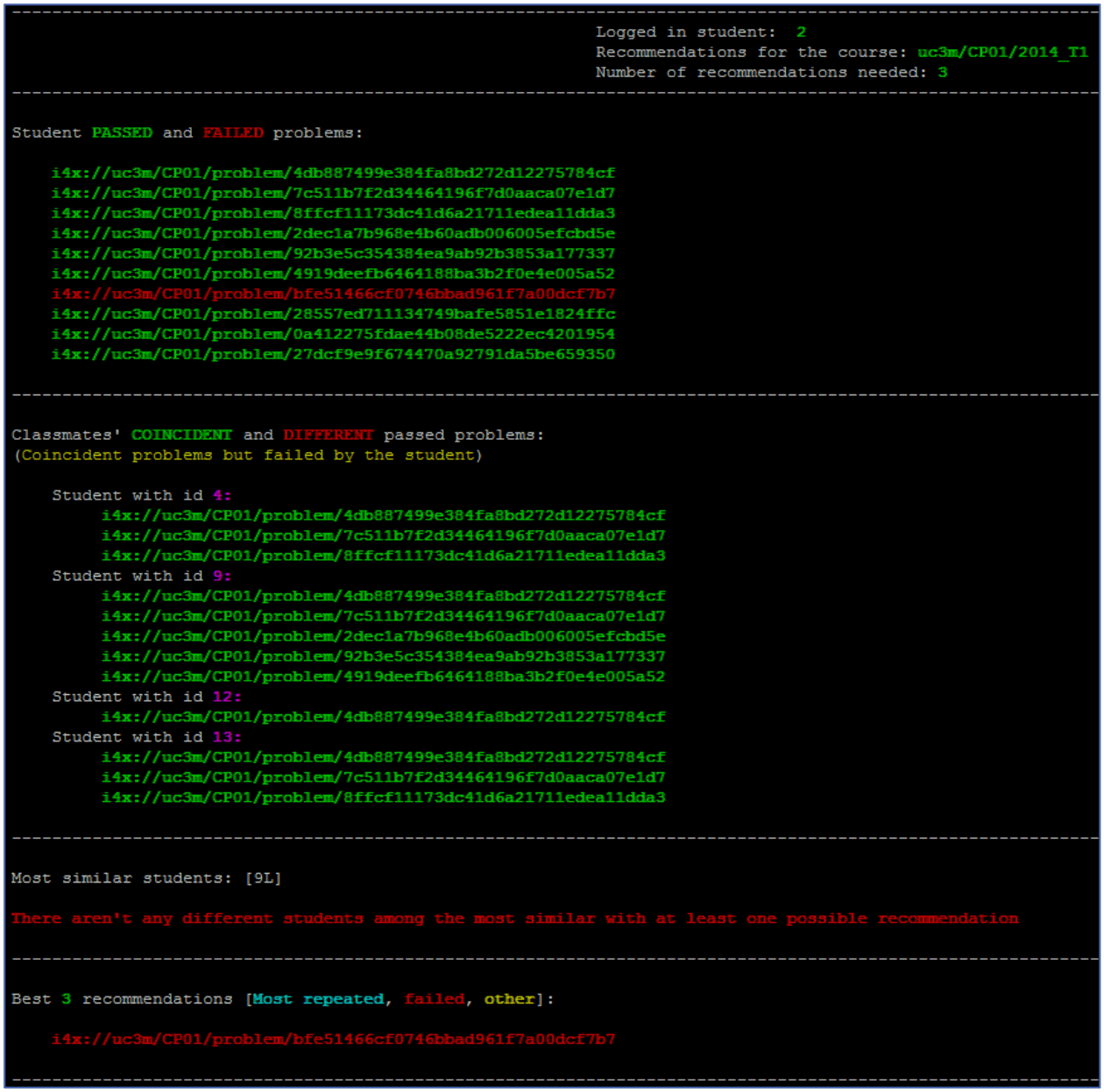


FIGURA 6-10. MENSAJES EN CONSOLA EN EL CASO 4

6.5. Caso 5: Varios alumnos registrados. El actual más atrasado

Elegimos otro alumno, uno que vaya más atrasado que el alumno al que le hemos estado haciendo el seguimiento hasta ahora, y vemos qué problemas se le recomiendan en la imagen 6-11.

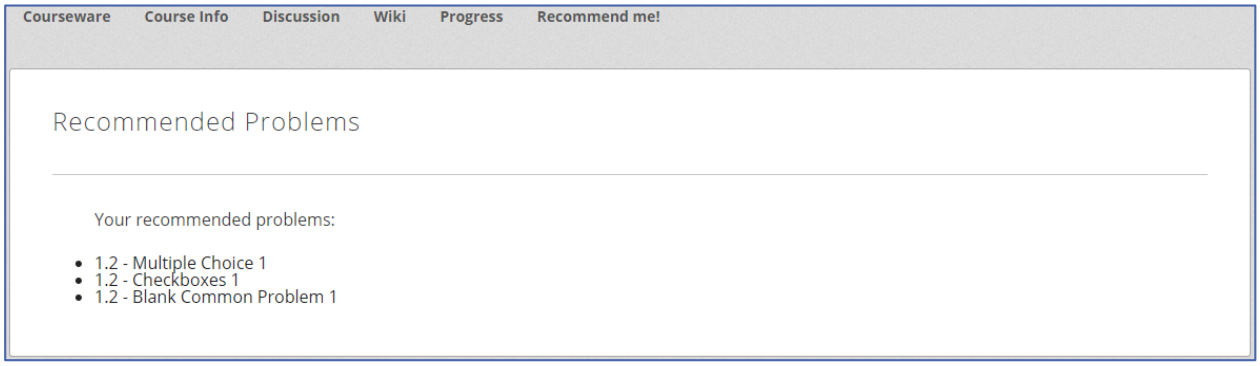


FIGURA 6-11. PROBLEMAS RECOMENDADOS PARA EL ALUMNO EN EL CASO 5

Observamos la tabla, en este caso se han creado las nuevas entradas para este alumno en este curso, como se muestra en la imagen 6-12.

```
mysql> select * from recommender_student;
+----+---------+------------------+-----------------------------------------------------------------+
| id | user_id | course_id        | module_id                                                       |
+----+---------+------------------+-----------------------------------------------------------------+
| 73 |       2 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/bfe51466cf0746bbad961f7a00dcf7b7 |
| 74 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
| 75 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
| 76 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
| 77 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7 |
| 78 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3 |
| 79 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e |
| 80 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337 |
+----+---------+------------------+-----------------------------------------------------------------+
8 rows in set (0.00 sec)
```

FIGURA 6-12. ESTADO DE LA TABLA RECOMMENDER_STUDENT EN EL CASO 5

```
                                                                Logged in student:  12
                                                                Recommendations for the course: uc3m/CP01/2014_T1
                                                                Number of recommendations needed: 3

Student PASSED and FAILED problems:

    i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf

Classmates' COINCIDENT and DIFFERENT passed problems:
(Coincident problems but failed by the student)

    Student with id 2:
        i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
        i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
        i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
        i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e
        i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337
        i4x://uc3m/CP01/problem/4919deefb6464188ba3b2f0e4e005a52
        i4x://uc3m/CP01/problem/28557ed711134749bafe5851e1824ffc
        i4x://uc3m/CP01/problem/0a412275fdae44b08de5222ec4201954
        i4x://uc3m/CP01/problem/27dcf9e9f674470a92791da5be659350
    Student with id 4:
        i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
        i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
        i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
    Student with id 9:
        i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
        i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
        i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e
        i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337
        i4x://uc3m/CP01/problem/4919deefb6464188ba3b2f0e4e005a52
    Student with id 13:
        i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
        i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
        i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3

Most similar students: [2L, 4L, 9L, 13L]
Least different students among the most similar with at least one possible recommendation: [4L, 13L]

Possible recommendations from each classmate:
    Student with id 4:
        i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
        i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
    Student with id 13:
        i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
        i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3

Number of times each problem is repeated:
    i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7: 2
    i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3: 2

Best 3 recommendations [Most repeated, failed, other]:

    i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
    i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
    i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e
```

FIGURA 6-13. MENSAJES EN CONSOLA EN EL CASO 5

Analizamos la información obtenida en consola y representada en la figura 6-13:

- En este caso estudiamos el alumno con `user_id = 12` y vemos que solo ha realizado y aprobado un problema.
- Se muestran los problemas en los que coincide (verde) y en los que difiere (rojo) con cada compañero en ese momento.
- En este caso coincide con todos en un problema (el único que ha realizado) pero con algunos difiere menos que con otros. Se cogen los alumnos menos diferentes de entre los más coincidentes (4 y 13).
- Se cuentan las veces que se repite cada posible recomendación (los problemas en rojo de los más coincidentes y menos diferentes) y se recomiendan los más repetidos.
- En este caso no tiene problemas suspensos (que se recomendarían primero), luego solo se recomiendan problemas por parte de los compañeros. En este caso solo hay dos problemas por repetición, por lo que se tomarán el resto de problemas de los problemas aprobados por otro de los compañeros más coincidentes (amarillo).

6.6. Caso 6: Varios alumnos registrados. Un solo compañero más coincidente

En este caso hacemos la prueba para un alumno que se encuentre en la media, es decir, que no sea de los más atrasados ni de los más adelantados en el curso. En la imagen 6-14 se muestra cada paso del algoritmo y se explican a continuación:

- El alumno ha suspendido un problema.
- Coincide con el alumno 2 en seis problemas, pero uno de ellos es el problema que ha suspendido, luego no se tiene en cuenta a la hora de buscar los compañeros más coincidentes. En este caso el compañero 2 es el más coincidente.
- Al tener un solo compañero más coincidente solo podemos tomarlo a él como compañero menos diferente.
- Difiere en tres problemas con el alumno actual (color rojo), y puesto que es el único compañero más coincidente y menos diferente solo se contabiliza una repetición de cada uno de ellos.
- Finalmente se muestran las recomendaciones, el problema suspenso del alumno y dos de los problemas del compañero ordenados según fueron creados. En la imagen 6-15 observamos cómo en la tabla `recommender_student` hay almacenadas cuatro recomendaciones, el problema suspenso y los tres recomendados por el alumno dos, pero solo se muestran tres de ellas, que son las requeridas.

```
------------------------------------------------------------------------------------------------------------------------
                                                                Logged in student:  9
                                                                Recommendations for the course: uc3m/CP01/2014_T1
                                                                Number of recommendations needed: 3
------------------------------------------------------------------------------------------------------------------------

Student PASSED and FAILED problems:

    i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
    i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
    i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
    i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e
    i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337
    i4x://uc3m/CP01/problem/4919deefb6464188ba3b2f0e4e005a52

------------------------------------------------------------------------------------------------------------------------

Classmates' COINCIDENT and DIFFERENT passed problems:
(Coincident problems but failed by the student)

    Student with id 2:
         i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
         i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
         i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
         i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e
         i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337
         i4x://uc3m/CP01/problem/4919deefb6464188ba3b2f0e4e005a52
         i4x://uc3m/CP01/problem/28557ed711134749bafe5851e1824ffc
         i4x://uc3m/CP01/problem/0a412275fdae44b08de5222ec4201954
         i4x://uc3m/CP01/problem/27dcf9e9f674470a92791da5be659350
    Student with id 4:
         i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
         i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
         i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
    Student with id 12:
         i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
    Student with id 13:
         i4x://uc3m/CP01/problem/4db887499e384fa8bd272d12275784cf
         i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7
         i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3

------------------------------------------------------------------------------------------------------------------------

Most similar students: [2L]
Least different students among the most similar with at least one possible recommendation: [2L]

Possible recommendations from each classmate:
    Student with id 2:
        i4x://uc3m/CP01/problem/28557ed711134749bafe5851e1824ffc
        i4x://uc3m/CP01/problem/0a412275fdae44b08de5222ec4201954
        i4x://uc3m/CP01/problem/27dcf9e9f674470a92791da5be659350

Number of times each problem is repeated:
    i4x://uc3m/CP01/problem/28557ed711134749bafe5851e1824ffc: 1
    i4x://uc3m/CP01/problem/0a412275fdae44b08de5222ec4201954: 1
    i4x://uc3m/CP01/problem/27dcf9e9f674470a92791da5be659350: 1

------------------------------------------------------------------------------------------------------------------------

Best 3 recommendations [Most repeated, failed, other]:

    i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3
    i4x://uc3m/CP01/problem/28557ed711134749bafe5851e1824ffc
    i4x://uc3m/CP01/problem/0a412275fdae44b08de5222ec4201954

------------------------------------------------------------------------------------------------------------------------
```

FIGURA 6-14. MENSAJES EN CONSOLA DEL CASO 6

```
mysql> select * from recommender_student;
+----+---------+-------------------+-----------------------------------------------------------------+
| id | user_id | course_id         | module_id                                                       |
+----+---------+-------------------+-----------------------------------------------------------------+
| 73 |       2 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/bfe51466cf0746bbad961f7a00dcf7b7 |
| 74 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
| 75 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
| 76 |       2 | uc3m/CP01/2014_T1 | NULL                                                            |
| 77 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7 |
| 78 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3 |
| 79 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e |
| 80 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337 |
| 81 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3 |
| 82 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/28557ed711134749bafe5851e1824ffc |
| 83 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/0a412275fdae44b08de5222ec4201954 |
| 84 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/27dcf9e9f674470a92791da5be659350 |
+----+---------+-------------------+-----------------------------------------------------------------+
12 rows in set (0.01 sec)
```

FIGURA 6-15. ESTADO DE LA TABLA RECOMMENDER_STUDENT PARA EL CASO 6

6.7. Caso 7: Alumno registrado en varios cursos

En este caso vamos a comprobar el funcionamiento del algoritmo en caso de que el alumno, tanto el actual como alguno de sus compañeros, esté registrado en más de un curso. Solo se obtienen de la tabla `courseware_studentmodule` los problemas correspondientes al curso en el que el alumno esté pidiendo recomendaciones.

Lo primero que debemos hacer es registrar al alumno en un nuevo curso. Para ello creamos un nuevo curso de prueba, CP02. Después resolvemos algunos problemas y miramos nuestros problemas recomendados. Hemos utilizado el mismo alumno que en el Caso 5. En la imagen 6-16 observamos lo siguiente:

- En la esquina superior derecha vemos que ha cambiado el identificador del curso.
- Se muestran los tres problemas realizados por el usuario. Ha suspendido uno de ellos.
- Solo se recomienda el problema suspenso ya que es el único alumno registrado en el curso y no se pueden hacer recomendaciones de otros compañeros.

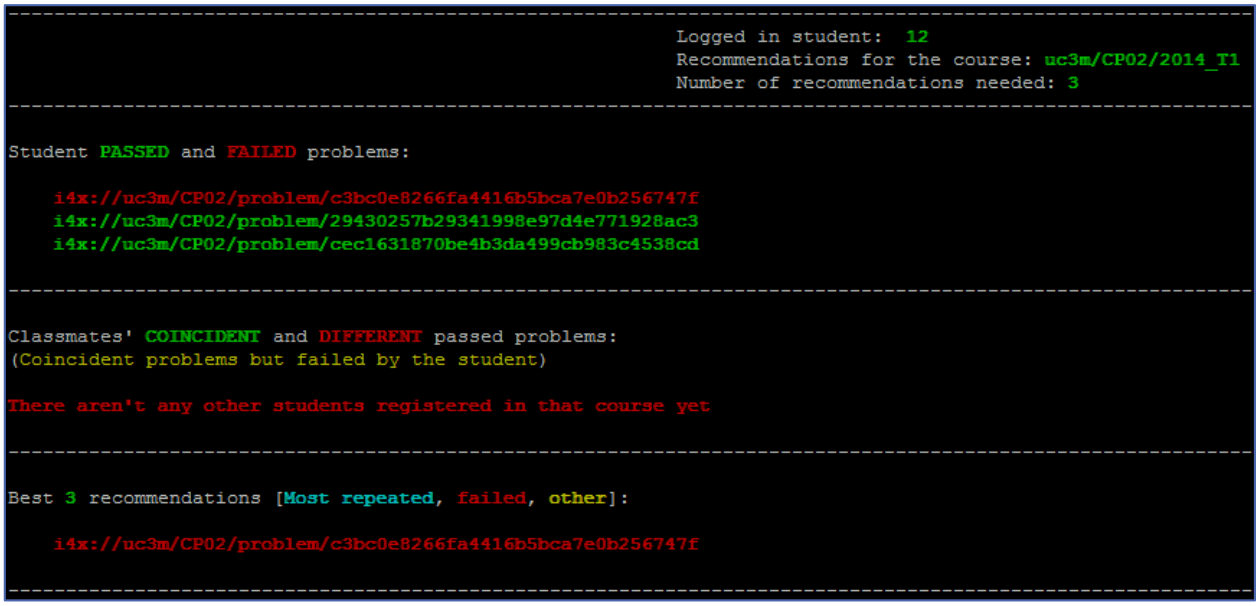


FIGURA 6-16. MENSAJES EN CONSOLA DEL CASO 7

Ahora miramos la tabla `recommender_student` (imagen 6-17) y vemos cómo se guardan las nuevas recomendaciones del alumno para este curso.

```
mysql> select * from recommender_student;
+----+---------+------------------+-----------------------------------------------------------+
| id | user_id | course_id        | module_id                                                 |
+----+---------+------------------+-----------------------------------------------------------+
| 73 |       2 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/bfe51466cf0746bbad961f7a00dcf7b7 |
| 74 |       2 | uc3m/CP01/2014_T1 | NULL                                                      |
| 75 |       2 | uc3m/CP01/2014_T1 | NULL                                                      |
| 76 |       2 | uc3m/CP01/2014_T1 | NULL                                                      |
| 77 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/7c511b7f2d34464196f7d0aaca07e1d7 |
| 78 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3 |
| 79 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/2dec1a7b968e4b60adb006005efcbd5e |
| 80 |      12 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/92b3e5c354384ea9ab92b3853a177337 |
| 81 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/8ffcf11173dc41d6a21711edea11dda3 |
| 82 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/28557ed711134749bafe5851e1824ffc |
| 83 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/0a412275fdae44b08de5222ec4201954 |
| 84 |       9 | uc3m/CP01/2014_T1 | i4x://uc3m/CP01/problem/27dcf9e9f674470a92791da5be659350 |
| 85 |      12 | uc3m/CP02/2014_T1 | i4x://uc3m/CP02/problem/c3bc0e8266fa4416b5bca7e0b256747f |
| 86 |      12 | uc3m/CP02/2014_T1 | NULL                                                      |
| 87 |      12 | uc3m/CP02/2014_T1 | NULL                                                      |
| 88 |      12 | uc3m/CP02/2014_T1 | NULL                                                      |
+----+---------+------------------+-----------------------------------------------------------+
16 rows in set (0.00 sec)
```

FIGURA 6-17. ESTADO DE LA TABLA RECOMMENDER_STUDENT PARA EL CASO 7

7. Gestión del proyecto

7.1. Planificación

Para la planificación de este proyecto se ha realizado un diagrama de Gantt, que permite representar de forma gráfica todo el proceso del proyecto y su avance en el tiempo. Para ello se ha utilizado el programa *GanttProject*.

De forma general, en el eje vertical del diagrama se representan las tareas que se han llevado a cabo para la realización del proyecto y en el eje horizontal el tiempo estimado requerido por cada una de ellas.

En la figura 7-1 se muestra el diagrama de Gantt del presente proyecto. La duración de cada tarea es aproximada, ya que no se ha llevado un recuento exhaustivo de los días dedicados a cada una. Según el mismo, desde que se propuso el proyecto hasta que se finalizó han transcurrido 145 días y se ha trabajado una media de 6 horas diarias, 870 horas en total.

Observamos que la 'Fase de Redacción' se extiende prácticamente a lo largo de todo el proyecto. Esto se debe a que tanto la plataforma edX, como el lenguaje de programación Python, así como el resto de herramientas y tecnologías utilizadas eran completamente nuevos para mí, por lo que he tenido que estar constantemente siguiendo tutoriales y buscando información al respecto (tarea de 'Documentación'). De la misma forma he ido realizando anotaciones y documentando las soluciones a los problemas encontrados durante todo el proyecto para incluirlas en la memoria posteriormente.

No se han representado en el diagrama de Gantt las reuniones con el tutor, pero se consideran dentro de las 870 horas de trabajo realizadas.

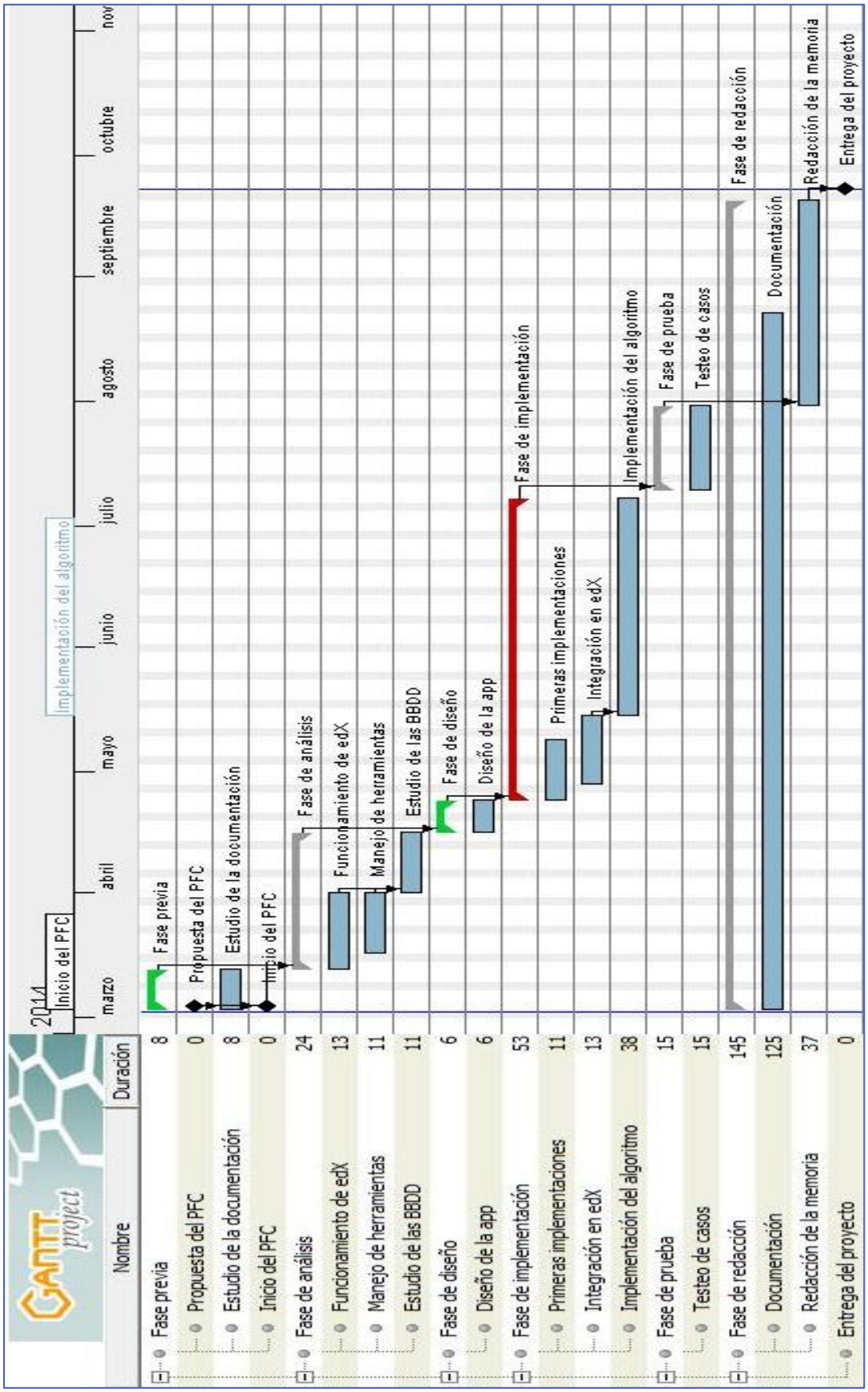


FIGURA 7-1. DIAGRAMA DE GANTT DEL PFC

7.2. Presupuesto

En este apartado se hace un desglose del presupuesto del proyecto. Todos los costes son calculados sin IVA.

7.2.1. Coste de personal:

Se tienen en cuenta las horas dedicadas al proyecto diferenciadas por fases, con el correspondiente sueldo según el cargo y las horas dedicadas por parte del tutor, tanto en las reuniones como en la lectura de la memoria. Se representa en la tabla 7-1 y hay que considerar lo siguiente:

- Las horas de la 'Fase de inicio' están incluidas en las horas trabajadas por el analista.
- Para la tarea de 'Documentación' se ha empleado una media de una hora diaria en lugar de seis (125 horas), pero para la tarea de 'Redacción de la memoria' se han empleado seis horas diarias (222 horas).

TABLA 7-1. COSTE DE PERSONAL

Cargo	Horas trabajadas		Coste por hora	Total
Analista	192		30€	5.760€
Diseñador	36		35€	1.260€
Desarrollador	318		22€	6.996€
Calidad y pruebas	90		30€	2.700€
Documentación	347		16€	2.320€
Jefe de proyecto	Reuniones	16	40€	5.552€
	Gestión de proyecto	20		
TOTAL				**24.588€**

7.2.2. Coste de hardware:

Se ha utilizado un portátil Lenovo con un procesador Intel Core i7 valorado en 599€. Aplicando la fórmula de la amortización siguiente obtenemos el resultado de la tabla 7-2:

$$A = \frac{t * C * u}{V}$$

Siendo:

- **t**: tiempo que ha transcurrido desde que fue adquirido
- **C**: coste del activo tangible
- **u**: porcentaje de uso dedicado al proyecto
- **V**: vida útil del activo

TABLA 7-2. COSTE DE HARDWARE

Descripción	Tiempo (meses)	Coste	% de uso	Vida útil (meses)	Coste imputable
Lenovo IdeaPad G510	6	599€	80	60	47,92€
TOTAL					**47,92€**

7.2.3. Coste de software y licencias

En la tabla 7-3 se detallan los costes del software utilizado. Se ha intentado en la medida de lo posible utilizar programas de código abierto para abaratar costes.

TABLA 7-3. COSTE DE SOFTWARE Y LICENCIAS

Descripción	Coste	Total
Microsoft Office 2013	99,99€	99,99€
Pacestar UML Diagrammer	108,28€	108,28€
Editor Komodo	0	0
GanttProject	0	0
TOTAL		**208,27€**

7.2.4. Coste de material fungible:

En la tabla 7-4 se detallan los costes de los materiales utilizados. En material de oficina se incluyen los folios, lápices, cuadernos, bolígrafos, etc. que han sido necesarios.

TABLA 7-4. COSTE DE MATERIAL FUNGIBLE

Descripción	Cantidad	Coste	Total
CD/DVD Vírgen	2	0,50€	1,0€
Material de oficina	1	20€	20€
TOTAL			**21€**

7.2.5. Coste total

En la tabla 7-5 se calcula el coste total del proyecto al que le hemos añadido un 5% perteneciente a los Costes Indirectos, que son aquellos que no están vinculados directamente con el proyecto pero este no se hubiera podido llevar a cabo sin ellos, como

puede ser la luz, el mantenimiento de las instalaciones donde se ha desarrollado, el sueldo del personal de administración, etc.

TABLA 7-5. COSTE TOTAL DEL PROYECTO

Descripción	Coste
Personal	24.588€
Amortización de Hardware	47,92€
Software y Licencias	208,27€
Material fungible	21€
Costes Indirectos (5%)	4.973€
TOTAL	**29.838,19€**

Si al presupuesto total del proyecto le añadimos el 21% de IVA asciende a **TREINTA Y SEIS MIL CIENTO CUATRO CON 21 EUROS**.

8. Conclusiones y trabajos futuros

8.1. Conclusiones

Se ha cumplido con el objetivo de este proyecto, el desarrollo de un recomendador de recursos para la plataforma edX. Para ello se ha diseñado un algoritmo de recomendación basado en las puntuaciones obtenidas en los problemas por el resto de los compañeros de curso. Este recomendador permite dar a conocer a los alumnos los siguientes problemas a realizar.

Basándonos en el apartado 2.4, donde se explican los distintos tipos de sistemas de recomendación, podemos decir que el recomendador desarrollado es una variación de un sistema de recomendación de tipo colaborativo, en el que los usuarios X e Y (siendo X e Y los compañeros de curso del alumno logueado en ese momento) se comportan de forma similar ante los n problemas del curso. La recomendación se realiza en dos pasos, primero se buscan aquellos compañeros cuyo patrón es similar al del alumno actual, es decir, aquellos que coinciden en más problemas aprobados y tienen menos problemas diferentes, y después se calcula una predicción en función de la popularidad de esos problemas diferentes. En resumen, se hacen predicciones sobre los problemas más adecuados para el alumno en ese momento mediante la recopilación de los problemas de los compañeros, convertidos en colaboradores, de forma que se intentan recomendar problemas que otros alumnos han resuelto correctamente.

Esta aplicación de recomendador es una herramienta opcional de ayuda para los alumnos que requieran cierta orientación durante la realización de un curso. Es el alumno el que decide hacer uso o no de la herramienta, ya que por un lado el algoritmo solo se ejecuta en caso de que se seleccione la pestaña *Recommend Me*! y por otro lado el alumno no está obligado a hacer caso a las recomendaciones de la herramienta desarrollada, pudiendo decidir hacer uso o no de las recomendaciones.

En relación al algoritmo de recomendación podemos decir que tiene algunas debilidades como las siguientes:

- Al estar basado en los problemas más populares de entre los alumnos más similares, podría haber problemas que nunca lleguen a ser recomendados. Esto puede ocurrir, por ejemplo, con los problemas de elevado nivel de dificultad, ya que en esos casos la tasa de aprobados es muy baja, por lo que sus índices de popularidad serán cercanos a cero y no serán propuestos.

- Otra de estas debilidades queda representada en los casos 1, 2 y 3 del capítulo 6. En un curso que ha comenzado recientemente, el número de alumnos registrados, o el número de problemas realizados por dichos alumnos, puede que sea bajo y no se puedan obtener las recomendaciones necesarias. Esto queda subsanado, en parte, por la recomendación de los problemas suspensos del alumno actual, que constituyen la primera opción de recomendación, ya que se considera lógico que para que un alumno pueda seguir avanzando en el curso deba haber aprobado todos los problemas anteriores.

De entre las tareas soportadas por los sistemas de secomendación en TEL actuales, enumeradas en la tabla 2-4, nuestro sistema de recomendación cubre las siguientes:

- ***Find peers***: se exploran los atributos de la tabla `courseware_studentmodule` con el objetivo de calcular la similitud entre el alumno actual y cada uno de sus compañeros de curso. De los compañeros más relevantes se obtendrán las posibles recomendaciones.

- ***Recommend sequence***: se recomienda una secuencia de recursos, en este caso solo de problemas, con la finalidad de alcanzar un objetivo de aprendizaje. Se buscan aquellos problemas realizados correctamente por un mayor número de compañeros similares y se recomiendan.

Para poder sacar conclusiones fiables es necesario probar el recomendador con alumnos reales interaccionando en un curso creado con diferentes recursos.

8.2. Trabajos futuros

El trabajo futuro más importante y necesario es evaluar la efectividad del algoritmo. Para ello será necesario aplicar alguna de las técnicas de evaluación explicadas en el apartado 2.6. También deberá seguirse el modelo de Kirckpatrick con el fin de estudiar la reacción de los usuarios, el aprendizaje, el comportamiento y los resultados. Las pruebas de escalabilidad son igualmente necesarias. Hasta que no se hayan llevado a cabo estas tareas de evaluación no se pueden sacar conclusiones y para ello será necesario implantar la aplicación en un escenario de prueba y recoger los resultados obtenidos.

Partiendo de este proyecto pueden implementarse otros algoritmos de filtrado colaborativo, adaptados igualmente a la plataforma edX, mediante los que se recomienden otro tipo de recursos, no solo problemas. Una nueva aplicación en la que se recojan votaciones u opiniones de los compañeros de curso acerca de la utilizad de un recurso, véase vídeo, audio, texto, etc.

Una de las nuevas funcionalidades que se pueden añadir a esta aplicación es el desarrollo de una interfaz gráfica con el fin de que el creador del curso pueda elegir en Studio entre las siguientes opciones:

- Que la pestaña *Recommend Me!* se muestre entre las otras pestañas del curso, es decir, que el creador decida si se activan o no las recomendaciones para ese curso.
- En caso de que se activen las recomendaciones dar la opción de elegir el número de recomendaciones que se van a almacenar en la tabla `recommender_student` de la base de datos de *MySQL*. Establecer límites.
- Igualmente, dar la opción de elegir el número de recomendaciones que le serán mostradas al alumno en pantalla. Este número debe de ser menor o igual que el número de recomendaciones almacenadas en la tabla.

Otra de las posibles mejoras futuras es que al seleccionar la pestaña de recomendación aparezcan los problemas recomendados del alumno en forma de secuencia, es decir, en una barra superior donde se puedan ir seleccionando y resolviendo directamente en lugar de mostrar simplemente el nombre del problema y que el alumno

tenga que buscarlo él mismo en el curso. En caso de que esto no fuera posible añadir links directos a los problemas recomendados en la pestaña de recomendación.

En marzo de 2014, los desarrolladores de edX presentaron un nuevo componente en la arquitectura de edX, los XBlocks[18]. Los XBlocks son componentes independientes e integrables entre sí creados por la comunidad de desarrolladores de edX con el fin de añadir nuevas funcionalidades y mejoras a los cursos, haciéndolos más atractivos para los alumnos. El recomendador desarrollado en este proyecto podría ser convertido en un XBlock en un trabajo futuro para evitar la modificación del propio código de la plataforma.

Inicialmente este proyecto se concibió como un proyecto de web semántica aplicado a la educación [104], en el que se definirían distintas ontologías y las recomendaciones también utilizarían metadatos asociados a los distintos recursos del curso para tomar las decisiones de los recursos a recomendar. La dificultad de la integración del código en la plataforma edX ya era lo suficientemente elevada y se descartó esta idea, dejándola para un posible trabajo futuro.

[18] https://xblock.readthedocs.org/en/latest/

A. ANEXO: Manual de instalación de la aplicación

En este anexo se explica qué ficheros hay que añadir y las modificaciones de código que hay que realizar para integrar la aplicación desarrollada en la plataforma edX.

1. El grueso del código de nuestra aplicación se encuentra en la carpeta 'recommender'. Es aquí donde están los ficheros que, como dijimos en el apartado 3.1.4, se generan de forma automática al crear una aplicación para un proyecto de *Django* ('_init_.py', 'models.py', 'tests.py') y el fichero 'data.py', donde se definen las funciones que recuperación de datos de *MongoDB* y *MySQL* y el algoritmo de recomendación. Situaremos esta carpeta en:

   ```
   /vagrant/edx-platform/lms/djangoapps/
   ```

2. Activamos nuestra aplicación para la instancia de *Django* modificando el fichero:

   ```
   /edx/app/edxapp/edx-platform/lms/envs/common.py
   ```

   Añadiendo 'recommender' a la lista de aplicaciones almacenadas en el parámetro `INSTALLED_APPS`.

3. Debemos añadir la *URL* que apunta a nuestra aplicación, para ello accedemos al fichero `/edx/app/edxapp/edx-platform/lms/urls.py` y añadimos al parámetro `urlpatterns` lo siguiente:

   ```
   url(r'^recommender/, include('recommender.urls'))
   ```

4. En este punto debemos comunicarle a Django que hemos realizado algunos cambios y que queremos utilizar la nueva aplicación. Para ello utilizamos los siguientes comandos en la consola:

   1. `edxapp@precise64:~/edx-platform$./manage.py lms schemamigration recommender --initial --settings=devstack`

      De esta forma se almacenan los cambios como una `migration`.

2. `edxapp@precise64:~/edx-platform$./manage.py lms schemamigration recommender --initial --settings=devstack`

   Se aplica la `migration` obtenida con el commando anterior y se crea en la base de datos la tabla 'recommender_student' que se corresponde con nuestro modelo 'recommender.Student'. Observamos que en la carpeta 'recommender' se ha creado una nueva carpeta llamada 'migrations', donde se van guardando las modificaciones realizadas.

5. El fichero 'recommendations.html' lo añadimos en la siguiente carpeta:

   `/vagrant/edx-platform/lms/templates/courseware/`

   Contiene el código *HTML* de la aplicación que se mostrará al alumno en el navegador al seleccionar la pestaña *Recommend Me!.*

6. Para que aparezca dicha pestaña (*Recommend Me!*) en el 'courseware' de la plataforma y funcione adecuadamente al seleccionarla hay que hacer lo siguiente:

   1. Acceder al CMS Studio como administrador, crear una Static Tab y darla el nombre deseado de la siguiente forma: Static Tab > Settings > Display Name: *Recommend Me!.*

   2. Acceder al fichero 'views.py' situado en:

      `/vagrant/edx-platform/lms/djangoapps/courseware/`

      Y referenciar el fichero *HTML* de nuestra aplicación en la función `static_tab(request, course_id, tab_slug)` como se muestra en la imagen A-1.

```
@ensure_csrf_cookie
def static_tab(request, course_id, tab_slug):
    """
    Display the courses tab with the given name.

    Assumes the course_id is in a valid format.
    """
    course = get_course_with_access(request.user, course_id, 'load')

    tab = tabs.get_static_tab_by_slug(course, tab_slug)
    if tab is None:
        raise Http404

    contents = tabs.get_static_tab_contents(
        request,
        course,
        tab
    )
    if contents is None:
        raise Http404
    staff_access = has_access(request.user, course, 'staff')
    return render_to_response('courseware/recommendations.html',
                              {'course': course,
                               'tab': tab,
                               'tab_contents': contents,
                               'staff_access': staff_access, })
```

FIGURA A-1. MODIFICACIÓN DE COURSEWARE/VIEWS.PY

B. ANEXO: Manual de utilización de la aplicación

En este anexo se explican los pasos que hay que seguir para ejecutar y utilizar la aplicación desarrollada en este proyecto.

En este caso se ha instalado la instancia de la plataforma edX en un servidor externo con SO Ubuntu 12.04 LTS. Para conectarnos al mismo desde Windows y tener acceso a todos los ficheros de la plataforma utilizamos el programa WinSCP[19] y para correr la instancia de la plataforma e interactuar con los ficheros de la misma utilizamos el programa PuTTY[20] (imagen B-1).

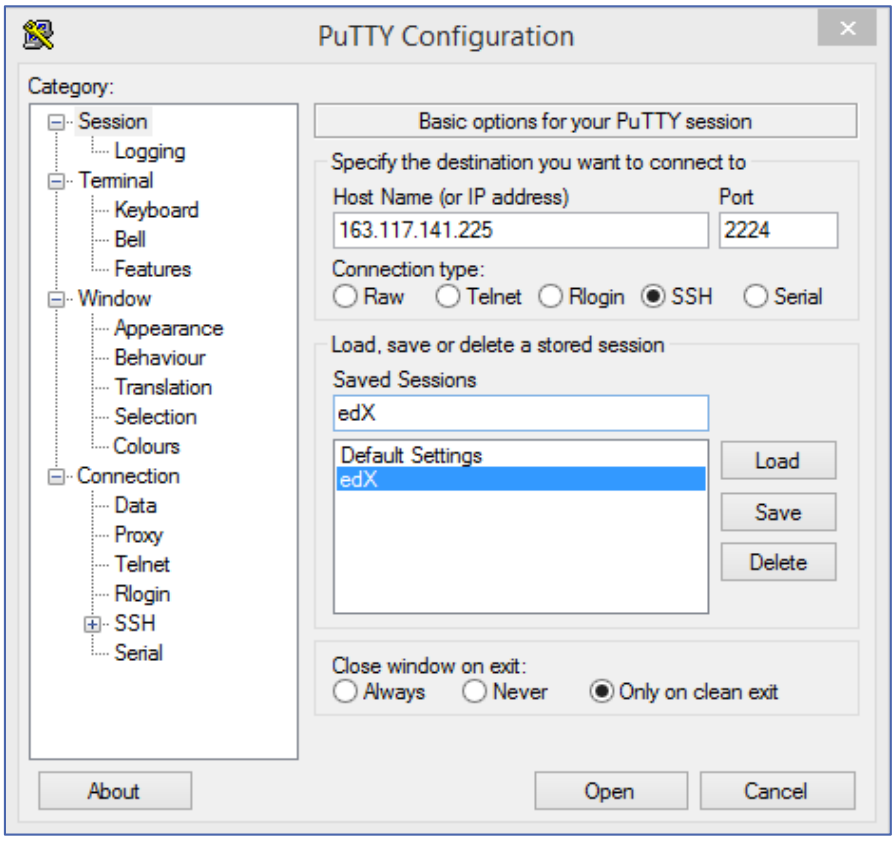


FIGURA B-1. PARÁMETROS DE CONEXIÓN CON EL SERVIDOR

[19] http://winscp.net/eng/download.php
[20] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Una vez introducidos los parámetros de conexión tanto en WinSCP como en PuTTY utilizamos el usuario y contraseña adjudicados por el administrador del servidor para poder acceder al mismo y después nos conectamos con el usuario 'edxapp' a l1a instancia de la plataforma con el siguiente comando:

```
sudo su edxapp.
```

Para correr la instancia de la plataforma utilizando la configuración de Developer Stack (devstack) introducimos los comandos mostrados en la imagen B-2:

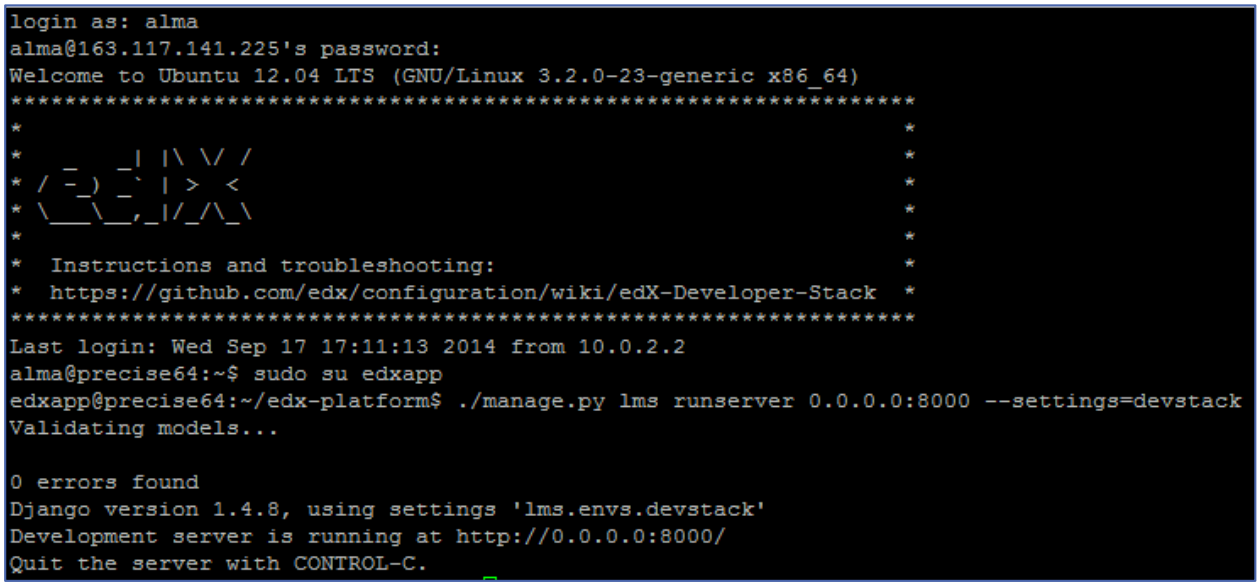


FIGURA B-2. ACCESO A LA INSTANCIA DE EDX

Una vez que tenemos corriendo la instancia de la plataforma vamos al navegador y accedemos al LMS de edX:

- Si se quiere acceder como alumno: `163.117.141.225:8000`
- Si se quiere acceder como administrador de *Django*: `163.117.141.225:8000/admin` (para ello hay que tener cuenta de administrador).

Para ejecutar las pruebas de evaluación de la aplicación es necesario entrar en el LMS con un perfil de alumno como hemos indicado y elegir la pestaña '*Recommend Me!*', de esta forma se obtienen las recomendaciones en el navegador y los mensajes de salida en consola. Se nos da información del identificador del alumno y del curso, del resto de

compañeros del curso, problemas coincidentes con el actual, problemas más populares, etc.

Si, por el contrario, somos un profesor con intención de crear un curso o gestionar alguno creado anteriormente debemos acceder a Studio, es decir, al CMS de edX. Para ello hay que utilizar el siguiente comando:

```
rake devstack[studio]
```

La dirección que debemos escribir en la barra de direcciones del navegador es:

```
http://163.117.141.225:8001/
```

Si necesitamos acceder a las bases de datos para ver tanto el contenido del curso (*MongoDB*) como toda la información en referencia a los alumnos registrados y su actividad (*MySQL*) debemos abrir otra sesión en PuTTY e introducir los siguientes comandos:

- Para acceder a *MySQL*:

```
edxapp@precise64:~/edx-platform$ sudo -u root mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 38
Server version: 5.5.35-0ubuntu0.12.04.2 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use edxapp;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

FIGURA B-3. ACCESO A MYSQL

Como se muestra en la imagen B-3 primero accedemos al gestor de la base de datos *MySQL* con permisos de administrador y posteriormente seleccionamos la

base de datos 'edxapp'. Para gestionar nuestra base de datos utilizamos el lenguaje estructurado *SQL*[21]

- Para acceder a *MongoDB*:

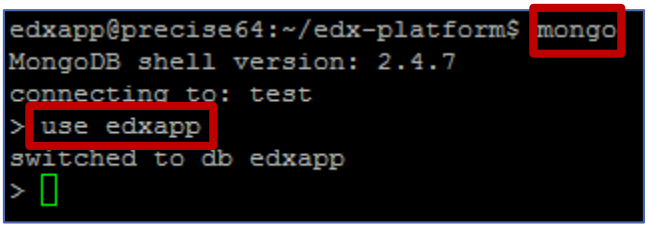


FIGURA B-4. ACCESO A MONGODB

Como se muestra en la imagen B-4 primero accedemos a *MongoDB* y seleccionamos la base de datos 'edxapp', donde se almacena toda la información referente al curso. El lenguaje utilizado para realizar las consultas y recuperar los documentos de las colecciones es el propio de *MongoDB*[22].

[21] http://www.w3schools.com/sql/
[22] http://docs.mongodb.org/manual/tutorial/query-documents/

Bibliografía

1. AZCORRA, A.; BERNARDOS, CARLOS JESÚS; GALLEGO, ÓSCAR [*ET AL.*] (2001): *Informe sobre el estado de la teleeducación en España.* [Informe en línea] Universidad Carlos III.
2. JOLLIFE, A.; RITTER, J.; STEVENS, D. (2001): *The online learning handbook*. Londres: Kogan Page.
3. CABERO, J.; GISBERT, M. (2005): *Formación en Internet. Guía para el diseño de materiales didácticos*. Sevilla: MAD.
4. PALLOF, R.; PRATT, K. (2003): *The virtual student*. San Francisco: Jossey Bass Wiley. Pallof *et al.* (2003, pág.130-131)
5. GISBERT, M.; CABERO, J.; CASTAÑO, C. [*ET AL.*] (2004): *Netlab: teleobservatorio universitario de docencia virtual*. Píxel-Bit. Revista de Medios y Educación. N.° 25, pág. 71-74.
6. CABERO, J. (2006): *Bases pedagógicas del e-learning*. Revista de Universidad y Sociedad del Conocimiento (RUSC) [artículo en línea]. Vol. 3, n.° 1. UOC.
7. SISTEMA DE GESTIÓN DE APRENDIZAJE. (22 de Junio de 2014). En Wikipedia, la enciclopedia libre. [artículo en línea] Recuperado el 30 de Julio de 2014 de http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_aprendizaje
8. CANELLA, A.: *CMS, LMS y LCMS*. Junio de 2011 [artículo en línea] http://www.noticiasusodidactico.com/blog/2011/06/cms-lms-y-lcms/
9. CLARENC, C. A.; CASTRO, S.M.; LÓPEZ DE LENZ, C.; MORENO, M.E.; TOSCO, N.B. (Diciembre, 2013): *Analizamos 19 plataformas de e-Learning: Investigación colaborativa sobre LMS*. Grupo GEIPITE, Congreso Virtual Mundial de e-Learning, www.congresoelearning.org
10. LUKES, D.: *What is and what is not a MOOC: A picture of family resemblance (working undefinition)*, 14 de Agosto de 2012. [artículo en línea] http://researchity.net/2012/08/14/what-is-and-what-is-not-a-mooc-a-picture-of-family-resemblance-working-undefinition-moocmooc/
11. UNIVERSITY BUSINESS: *What went wrong with AllLearn?* Junio de 2006. [artículo en línea] http://www.universitybusiness.com/article/what-went-wrong-alllearn
12. KATZ, D.: *Financial worries shut down AllLearn*, 20 de Marzo de 2006. [artículo en línea] http://yaledailynews.com/blog/2006/03/20/financial-worries-shut-down-alllearn/

13. HANE, P.J.: *Fathom this: academic and cultural institutions partner to create interactive knowledge company*. Information Today, 10 de Abril de 2000. [artículo en línea] http://newsbreaks.infotoday.com/nbreader.asp?ArticleID=17822

14. SIEMENS, G.: *What is the theory that underpins our moocs*, 3 de Junio de 2012. [artículo en línea] http://www.elearnspace.org/blog/2012/06/03/what-is-the-theory-that-underpins-our-moocs/

15. HSU, JEREMY: *Professor leaving Stanford for online education startup*, Enero de 2012. [artículo en línea] http://www.nbcnews.com/id/46138856/ns/technology_and_science-innovation/t/professor-leaving-stanford-online-education-startup/#.T-JhGWNSQWo

16. DOWNES, S.: *Connectivism and Connective Knowledge: Essays on meaning and learning networks* (2012) [artículo en línea] http://www.downes.ca/files/Connective_Knowledge-19May2012.pdf

17. SCOPEO (2013). SCOPEO INFORME Nº2. *MOOC: Estado de la situación actual, posibilidades, retos y futuro*. Salamanca: Universidad de Salamanca-Centro Internacional de Tecnologías Avanzadas.

18. VÁZQUEZ, E.; LÓPEZ, E.; SARASOLA, J.L. (2013): *La expansión del conocimiento abierto: los MOOC*. Barcelona: Octaedro.

19. DAVARA, F.: *La educación digital; Los MOOC (II),* 10 de abril de 2014. [artículo en línea] http://fernandodavara.com/la-educacion-digital-los-moocs-2/

20. LISA: *Three kinds of MOOCs*, 15 de Agosto de 2012. [artículo en línea] http://lisahistory.net/wordpress/2012/08/three-kinds-of-moocs/

21. KOVALCHIK, S.: *Udacity aims to reach 160.000+ students*. [artículo en línea] http://www.significancemagazine.org/details/webexclusive/2098135/Udacity-aims-to-teach-160000-students-statistics.html

22. GEE, S.: *MITx – the Fallout Rate*, 12 de Julio de 2012. [artículo en línea] http://www.i-programmer.info/news/150-training-a-education/4372-mitx-the-fallout-rate.html

23. QUILLEN, I.: *Why do students enroll in (but don't complete) MOOC courses?*, 5 de Abril de 2013. [artículo en línea] http://blogs.kqed.org/mindshift/2013/04/why-do-students-enroll-in-but-dont-complete-mooc-courses/

24. HILL, P.: *Insight on MOOC student types from ELI Focus Session*, 3 de Abril de 2013. [artículo en línea] http://mfeldstein.com/insight-on-mooc-student-types-from-eli-focus-session/

25. AFSHAR, V.: *Adoption of Massive Open Online Courses* (Worldwide Survey), 20 de Mayo de 2013. [artículo en línea] http://www.huffingtonpost.com/vala-afshar/infographic-adoption-of-m_b_3303789.html

26. NEWS OFFICE: *What is edX?*, 2 de Mayo de 2012. [artículo en línea] http://newsoffice.mit.edu/2012/edx-faq-050212

27. DIARIO LINUX: *Introducción a Open edX*, 21 de Mayo de 2014. [artículo en línea] http://diariolinux.com/2014/05/21/introduccion-a-open-edx/

28. EDX. (4 de Agosto de 2014). En Wikipedia, la enciclopedia libre. [artículo en línea] Recuperado el 10 de Agosto de 2014 de http://en.wikipedia.org/wiki/EdX#Partnerships_.26_Participating_institutions

29. ANDERSON, N.: *Harvard-MIT venture edX teams with Google on platform for free online courses*, 10 de Septiembre de 2013. [artículo en línea] http://www.washingtonpost.com/local/education/harvard-mit-venture-edx-teams-with-google-on-platform-for-free-online-courses/2013/09/10/02879cf4-1a13-11e3-8685-5021e0c41964_story.html

30. PEREIRA, J.A.: *Introducción a Open edX (II),* 27 de mayo de 2014. [artículo en línea] http://diariolinux.com/2014/05/27/introduccion-a-open-edx-ii/

31. JULIÁN MATEOS, M.; MUÑOZ MERINO, P. J.: *Guía edX*. Universidad Carlos III. Leganés, Diciembre de 2013.

32. RICCI, F.; ROKACH, L.; SHAPIRA, B.: *Introduction to Recommender Systems Handbook*, Springer, 2011, pp. 1-35

33. KRULWICH, B.: *Lifestyle finder: intelligent user profiling using large-scale demographic data.* Artificial Intelligence Magazine, vol. 18, no. 2, pp. 37–45, 1997.

34. BURKE, R.: *Hybrid recommender systems: survey and experiments. User Modelling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

35. GUTTMAN, R.H.: *Merchant differentiation through integrative negotiation in agent-mediated electronic commerce*, M.S. thesis, School of Architecture and Planning, MIT, 1998.

36. GOLDBERG, K.; ROEDER, T.; GUPTA, D.; PERKINS, C.: *Eigentaste: a constant time collaborative filtering algorithm*. Information Retrieval, vol. 4, no. 2, pp. 133–151, 2001.

37. MILLER, B.N.; KONSTAN, J.A.; RIEDL, J.: PocketLens: *toward a personal recommender system*. ACM Transactions on Information Systems, vol. 22, no. 3, pp. 437–476, 2004.

38. SARWAR, B.M.; KARYPIS, G.; KONSTAN, J.A.; RIEDL, J.: *Item-based collaborative filtering recommendation algorithms in Proceedings of the 10th International Conference on World Wide Web* (WWW '01)pp. 285–295, May 2001.

39. MCLAUGHLIN, M.R.; HERLOCKER, J.L.: *A collaborative filtering algorithm and evaluation metric that accurately model the user experience in Proceedings of 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '04), pp. 329–336, Sheffield, UK, 2004.

40. HERLOCKER, J.L.; KONSTAN, J.A.; TERVEEN, L.G.; RIEDL, J.T.: *Evaluating collaborative filtering recommender systems*. ACM Transactions on Information Systems, vol. 22, no. 1, pp. 5–53, 2004)

41. SALTON, G.; MCGILL, M.; *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, NY, USA, 1983.

42. KARYPIS, G.: *Evaluation of item-based top-N recommendation algorithms in Proceedings of the International Conference on Information and Knowledge Management* (CIKM '01), pp. 247–254, Atlanta, Ga, USA, November 2001.

43. PEARL, J.: *Probabilistic Reasoning in Intelligent Systems*: *Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, Calif, USA, 1988.

44. HAN, J.; KAMBER, M.: *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

45. VUCETIC, S.; OBRADOVIC, Z.: *Collaborative filtering using a regression-based approach*. Knowledge and Information Systems, vol. 7, no. 1, pp. 1–22, 2005.

46. ADOMAVICIUS, G.; TUZHILIN, A.: *Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions*. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749, 2005.

47. YU, K.; SCHWAIGHOFER, A.; TRESP, V.; XU, X.; KRIEGEL, H.P.: *Probabilistic memory-based collaborative filtering.* IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 1, pp. 56–69, 2004.

48. CLAYPOOL, M.; GOKHALE, A.; MIRANDA, T. [ET AL.]: *Combining content-based and collaborative filters in an online newspaper in Proceedings of the SIGIR Workshop on Recommender Systems:* Algorithms and Evaluation, Berkeley, Calif, USA, 1999.

49. MCCRAE, J.; PIATEK, A.; LANGLEY, A.: *Collaborative Filtering*.

50. RESNICK, P.; VARIAN, H.R.: *Recommender systems*. Communications of the ACM, vol. 40, no. 3, pp. 56–58, 1997.

51. FLEDER, D.; HOSANAGAR, K.: *Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity*. *Management Science*, mayo de 2009.

52. PAZZANI, M.J.: *A framework for collaborative, content-based and demographic filtering*. Artificial Intelligence Review, vol. 13, no. 5-6, pp. 393–408, 1999.

53. ADOMAVICIUS, G.; TUZHILIN, A.: *Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions*. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749, 2005.

54. BALABANOVIĆ, M.; SHOHAM, Y: *Content-based, collaborative recommendation.* Communications of the ACM, vol. 40, no. 3, pp. 66–72, 1997.

55. MANOUSELIS, N.; DRACHSLER, H.; VUORIKARI, R.; HUMMEL, H.; KOPER, R.*: Recommender Systems in Technology Enhanced Learning.*

56. BRUSILOVSKY, P. (2001): *Adaptive hypermedia*. *User Modeling and User-Adapted Interaction*, 11(1-2), 87–110.

57. RECKER, M.M.; WILEY, D.A.: *A non-authoritative educational metadata ontology for filtering and recommending learning objects*. Journal of Interactive Learning Environments, 9(3), 255-271, December 2001.

58. RECKER, M.M.; WILEY, D.A.: *An interface for collaborative filtering of educational resources in Proc. of the 2000 International Conference on Artificial Intelligence*, Las Vegas, USA, 26-29 June 2000.

59. RECKER, M.M.; WALKER, A.: *Supporting 'word-of-mouth' social networks via collaborative information filtering.* Journal of Interactive Learning Research, 14(1), pp. 79-98, 2003.

60. RECKER, M.; WALKER, A.; LAWLESS, K.: *What do you recommend? Implementation and analyses of collaborative filtering of Web resources for education*. Instructional Science, 31(4/5), 229-316, 2003.

61. ANDERSON, M.; BALL M.; BOLEY, H.; GREENE, S.; HOWSE, N.; LEMIRE, D.; MCGRATH, S.: RACOFI: *A Rule-Applying Collaborative Filtering System*. In Proc. IEEE/WIC COLA'03, Halifax, Canada, October 2003. Aroyo, L., Mizoguchi,

62. LEMIRE, D.; BOLEY, H.; MCGRATH, S.; BALL, M. (2005): *Collaborative Filtering and Inference Rules for Context-Aware Learning Object Recommendation*. International Journal of Interactive Technology and Smart Education, 2(3). Liber, O., Johnson, M.: Personal Learning Environments.

63. RAFAELI, S.; BARAK, M.; DAN-GUR, Y.; TOCH, E. (2004). *QSIA: a web-based environment for learning, assessing and knowledge sharing in communities*. Computers & Education, 43(3), 273-289.

64. RAFAELI, S.; DAN-GUR, Y.; BARAK, M. (2005): *Social Recommender Systems: Recommendations in Support of E-Learning*. International Journal of Distance Education Technologies, 3(2), 29-45.

65. AVANCINI, H.; STRACCIA, U.: *User recommendation for collaborative and personalised digital archives.* International Journal of Web Based Communities, 1(2), 163-175, 2005.

66. DRON, J.; MITCHELL, R.; BOYNE, C.; SIVITER, P. (2000a): *CoFIND: steps towards a self-organising learning environment*. Proceedings of WebNet 2000 (pp. 146-151), San Antonio, Texas, USA: AACE.

67. SHEN, L.; SHEN, R.: *Learning Content Recommendation Service based on Simple Sequencing Specification*, in Liu W. et al., (Eds.), ICWL 2004, LNCS 3143, 363-370, 2004.

68. TANG, T.; MCCALLA, G.: *Smart Recommendation for an Evolving E-Learning System*, in Proc. Of the Workshop on Technologies for Electronic Documents for Supporting Learning, International Conference on Artificial Intelligence in Education (AIED 2003), Sydney, Australia, July 20-24, 2003.

69. TANG, T.Y.; MCCALLA, G.I.: *Beyond Learners' Interest: Personalized Paper Recommendation Based on Their Pedagogical Features for an E-Learning System*, in Proc. of the 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2004), Auckland, New Zealand, August 9-13, 2004b.

70. TANG, T.Y.: MCCALLA, G.I.: *On the Pedagogically Guided Paper Recommendation for an Evolving Web-Based Learning System*, in Proc. of the 17th International FLAIRS Conference, Miami Beach, Florida, May 17-19, 2004a. 24 Nikos Manouselis1, Hendrik Drachsler2, Riina Vuorikari2,3, Hans Hummel2, Rob Koper2

71. TANG, T.Y.; MCCALLA, G.I.: *Smart Recommendation for an Evolving E-Learning System: Architecture and Experiment*, International Journal on E-Learning, 4 (1), 105-129, 2005.

72. TANG, T.Y.; MCCALLA, G.I.: *Utilizing Artificial Learner on the Cold-Start Pedagogical-Value based Paper Recommendation*, in Proc. of AH 2004: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Eindhoven, The Netherlands, August 23-26, 2004c. Terveen, L.

73. DRACHSLER, H.; HUMMEL, H.G.K.; VAN DEN BERG, B.; ESHUIS, J.; BERLANGA, A.; NADOLSKI, R.; WATERINK, W.; BOERS, N.; KOPER, R. (2009). *Effects of the ISIS Recommender System for navigation support in self-organized learning networks*. Journal of Educational Technology and Society.

74. MANOUSELIS, N.; VUORIKARI, R.; VAN ASSCHE, F.: *Simulated Analysis of MAUT Collaborative Filteringfor Learning Object Recommendation*, in Proc. of the Workshop on Social InformationRetrieval in Technology Enhanced Learning (SIRTEL 2007), Crete, Greece, 2007.

75. TSAI, K.H.; CHIU, T.K.; LEE, M.C.; WANG, T.I.: *A Learning Objects Recommendation Model based on the Preference and Ontological Approaches*, in Proc. Of of the 6th International Conference on Advanced Learnrning Technologies (ICALT'06), IEEE Computer Society, Los Alamitos, California,IEEE Computer Society Press, 2006.

76. FIAIDHI, J.: RECOSEARCH*: A Model for Collaboratively Filtering Java Learning Objects*, International Journal of Instructional Technology and Distance Learning, 1(7), 35-50, 2004.

77. NADOLSKI, R.; VAN DEN BERG, B.; BERLANGA, A.; DRACHSLER, H.; HUMMEL, H.G.K.; KOPER, R.; SLOEP, P. (2009*): Simulating light-weight Personalised Recommender Systems in learning networks: A case for Pedagogy-Oriented and Rating based Hybrid Recommendation Strategies*. Journal of Artificial Societies and Social Simulation (JASSS).

78. DRACHSLER, H.; PECCEU, D.; ARTS, T.; HUTTEN, E.; RUTLEDGE, L.; VAN ROSMALEN, P.; HUMMEL, H.G.K.; KOPER, R.: *ReMashed - Recommendations for Mash-Up Personal Learning Environments*. In: Cress, U., Dimitrova, V., Specht, M. (eds.): Learning in the Synergy of Multiple Disciplines, EC-TEL 2009 Vol. 5794. Springer Nice, France (2009a)

79. DRACHSLER, H.; PECCEU, D.; ARTS, T.; HUTTEN, E.; RUTLEDGE, L.; VAN ROSMALEN, P.; HUMMEL, H.G.K.; KOPER, R.: *ReMashed - An Usability Study of a Recommender System for Mash-Ups for Learning*. 1st Workshop on Mashups for Learning at the International Conference on Interactive Computer Aided Learning, Villach, Austria. (2009b)

80. KOUTRIKA, G.; IKEDA, R.; BERCOVITZ, B.; GARCIA-MOLINA, H.: *Flexible Recommendations over Rich Data*, in Proc. of the 2nd ACM International Conference on Recommender Systems (Rec-Sys'08), Lausanne, Switzerland, October 23-25, 2008.

81. KOUTRIKA, G.; IKEDA, R.; BERCOVITZ, B.; GARCIA-MOLINA, H.; LIOU, H.: CourseRank: *A Closed-Community Social System Through the Magnifying Glass*, in Proc. Of the 3rd International AAAI Conference on Weblogs and Social Media (ICWSM'09), San Jose, California, May 17– 20, 2009.

82. GOMEZ-ALBARRAN, M.; JIMENEZ-DIAZ, G.: *Recommendation and Students'Authoring in Repositories of Learning Objects: A Case-Based Reasoning Approach*, International Journal of Emerging Technologies in Learning (iJET), 4(1), 35-40, 2009.

83. AEHNELT, M.; EBERT, M.; BEHAM, G.; LINDSTAEDT, S.; PASCHEN, A.: *A Socio-technical Approach towards Supporting Intra-organizational Collaboration*, in Dillenbourg P. and Sprecht M., (Eds.), Proc. Of the 3rd European Conference on Technology Enhanced Learning (EC-TEL 2008), LNCS 5192, Berlin; Heidelberg; New York: Springer, 33-38, 2008.

84. SANTOS, O.C.: *A recommender system to provide adaptive and inclusive standard-based support along the elearning life cycle*, in Proc. of the 2008 ACM Conference on Recommender Systems, 319-322, Lausanne, Switzerland, 2008.

85. JANSSEN, J.; TATTERSALL, C.; WATERINK, W.; VAN DEN BERG, B.; VAN ES, R.; BOLMAN, C. [ET AL.] (2005): *Self-organising navigational support in lifelong learning: how predecessors can lead the way.* Computers & Education, 49, 781-793.

86. HUANG, D-S. [ET AL]. (EDS.): *ICIC 2009*, LNCS 5754, pp. 307–316, 2009.

87. KHRIBI, M.K.; JEMNI, M.; NASRAOUI, O. (2009): *Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval.* Educational Technology & Society, 12(4), 30–42.

88. BREESE, J. S.; HECKERMAN, D.; KADIE, C. (1998): *Empirical analysis of predictive algorithms for collaborative filtering*. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98). G. F. Cooper, and S. Moral, Eds. Morgan-Kaufmann, San Francisco,Calif., pp. 43-52, 1998.

89. HERLOCKER, J; KONSTAN, J.; TERVEEN, L.; RIEDL, J.:. *Evaluating collaborative filtering recommender systems*. In *ACM TOIS*, volume 22, pages 5.53. ACM Press, 2004.

90. YAO, Y.Y.: *Measuring retrieval effectiveness based on user preference of documents*, Journal of the American Society for Information Science, vol. 46, no. 2, pp. 133–145, 1995.

91. WEIBELZAHL, S.; PARAMYTHIS A. (EDS.) (2003): *2nd Workshop on Empirical Evaluation of Adaptive Systems*, International Conference on User Modeling, UM2003, Johnstown.

92. DJANGO (FRAMEWORK) (20 de julio de 2014). En Wikipedia, la enciclopedia libre. [artículo en línea] Recuperado el 14 de Agosto de 2014 de http://es.wikipedia.org/wiki/Django_(framework)

93. INFANTE MONTERO, S. (2012): *Curso Django: Instalación y primera aplicación.* [artículo en línea] http://www.maestrosdelweb.com/editorial/curso-django-instalacion-y-primera-aplicacion/

94. PALAT, J. (2012*): Introducing Vagrant*. [artículo en línea] http://www.linuxjournal.com/content/introducing-vagrant

95. ARTÍCULO EN LÍNEA: https://github.com/edx/configuration/wiki/edX-Developer-Stack

96. ARTÍCULO EN LÍNEA: http://code.edx.org/

97. MONGODB (3 de agosto de 2014). En Wikipedia, la enciclopedia libre. [artículo en línea] Recuperado el 12 de Agosto de 2014 de http://es.wikipedia.org/wiki/MongoDB

98. PARAMIO, C. (2011): *Una introducción a MongoDB*. [artículo en línea] http://www.genbetadev.com/bases-de-datos/una-introduccion-a-mongodb

99. CHODOROW, K; DIROLF, M. (September 23, 2010): *MongoDB: The Definitive Guide* (1st edición), O'Reilly Media.

100. MYSQL (3 de agosto de 2014). En Wikipedia, la enciclopedia libre. [artículo en línea] Recuperado el 20 de Agosto de 2014 de http://es.wikipedia.org/wiki/MySQL

101. SQL (12 de agosto de 2014). En Wikipedia, la enciclopedia libre. [artículo en línea] Recuperado el 20 de Agosto de 2014 de http://es.wikipedia.org/wiki/SQL

102. ARTÍCULO EN LÍNEA: http://edx.readthedocs.org/en/latest/internal_data_formats/sql_schema.html#user-data

103. ARTÍCULO EN LÍNEA: http://edx-partner-course-staff.readthedocs.org/en/latest/creating_content/create_problem.html

104. Muñoz Merino, P.J.; Delgado Kloos, C.: *An Architecture for Combining Semantic Web Techniques with Intelligent Tutoring Systems*. Universidad Carlos III. Madrid.